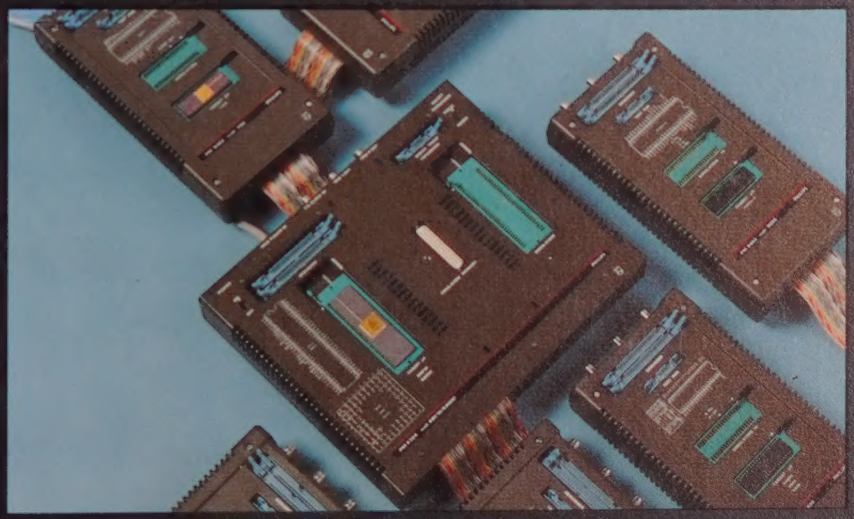
Test & Measurement

PHILIPS

ASSEMBLY DEBUGGER 68000/68010 HARDWARE AND SOFTWARE INSTALLATION

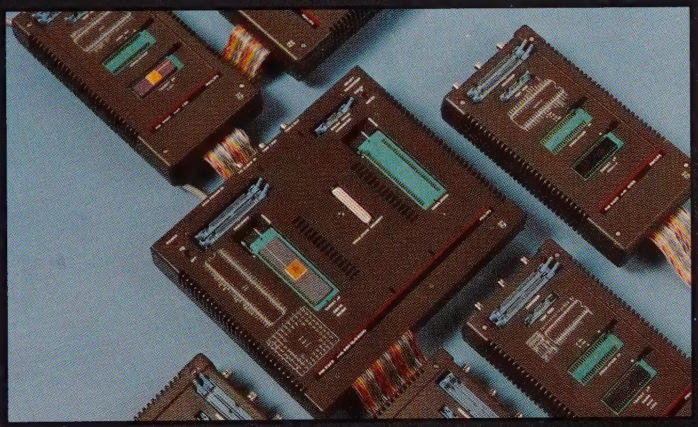


PMDS

Test & Measurement

PHILIPS

ASSEMBLY DEBUGGER 68000/68010



PMDS

A publication of

Philips I&E

T & M - DTS Documentation Department,

Eindhoven, The Netherlands

Publication number 4822 871 90784

Copyright © 1987 Philips Export BV

All rights reserved. Reproduction in whole or part is prohibited without the written consent of the copyright owner.

UNIX is a trademark of Bell Laboratories
IBM is a trademark of International Business Machines Corporation
PC–AT is a trademark of International Business Machines Corporation
XENIX is a trademark of Microsoft Corporation
Microsoft is a trademark of Microsoft Corporation

Printed in The Netherlands.

MANUAL PURPOSE AND STRUCTURE

PURPOSE

This manual is aimed at the new user to the Philips Microcomputer Development System 68010/68000 Assembly Debugger. It is assumed that the user is fully aware of how to use the PMDS and also is fully conversant with the 68010/68000 Microprocessors.

STRUCTURE

The manual contains six chapters that are organised into five parts, each part separated by a tab card. The first tab is marked **Software Reference** and contains Chapters 1, 2 and 3. The second tab card is marked **Commands** and contains Chapter 4. The third tab card is marked **Sample Run** and contains Chapter 5. The fourth tab card is marked **Hardware Reference** and contains Chapter 6. A sixth tab card marked **Appendixes** is also included to provide ease of access to Appendix A and Appendix B.

A summary of the contents of each chapter is provided below.

1. DEBUGGING FACILITIES

This chapter describes the full range of features available with the Assembly Debugger.

2. MENU BASED COMMANDS

This chapter explains how to use the menu based commands.

3. DIRECT MODE COMMANDS

This chapter explains how to use the commands available in direct input mode.

4. COMMAND DIRECTORY

This chapter provides an alphabetical listing of all the Debugger commands. It is not intended that this chapter is read in full, but referred to when specific information about each command is required.

5. DEBUGGING AN ASSEMBLY PROGRAM

This is a tutorial chapter of the manual, showing how to debug an assembly program.

6. HARDWARE REFERENCE GUIDE

This chapter contains the electrical specifications of the Debugger hardware and MAB. Also included is the emulation probe specification and the timing diagrams of the 68000.

APPENDIX A - FUNCTION KEYS

APPENDIX B - ASCII TABLE

INDEX

PROBLEM REPORT FORMS

SOFTWARE ON FLOPPY DISCS

RELEASE NOTES

SOFTWARE FEATURES

The screen oriented Debugger provides the user with a menu driven, window based screen interface, combining the functions of the Debugger with file management and operating system interface functions.

Several debugger features are summarised below. Refer to Chapters 1, 2 and 3 for a more detailed description of each feature.

symbolic debugging	symbolic names can be used to refer to program locations.
window management system	up to four windows, each with four levels, can be created.
auto-update	the contents of *auto-update* windows are automatically updated at each program break.
data formats	hexadecimal, decimal, octal and ascii formats may be used. In addition, the radix option assembly may be used.
monitors	software and hardware monitors are provided.

conditional commands	the *if* and *while* commands are used to execute debugger commands conditionally.
event counter	the real time event counter is a 24-bit counter which can count a wide variety of hardware events. Emulation breaks may be generated by the event counter reaching a certain value.
real time tracing	the real time trace mechanism can be used to trace the execution steps of a program.

Files can be edited, displayed and removed without leaving the Debugger. Special types of files can be created within the Debugger. *Files* commands are explained in Chapter 2.

log files	a command log file can be opened to record all debug commands issued during a session. A command output file can be opened to record all debugger output to the issued commands.
command files	instead of entering commands during a debugging session, they can be entered into a file. The command file may then be invoked during a debugging session.

Furthermore, help on how to use the commands can be achieved

at any time during a session.

CONTENTS

		Page
1	DEBUGGING FACILITIES	1-3
1.1	STARTING A DEBUGGING SESSION	1-3
1.1.1	Invoking the Debugger	1-3
1.1.2	The Command File .debugrc	1-4
1.1.3	Altering the Parameters EDITOR and VIEW	1-5
1.2	HELP FACILITY	1-6
1.3	WINDOWS	1-7
1.3.1	Menu Window	1-8
1.3.2	Pop-up Windows	1-9
1.3.3	Display Windows	1-9
1.4	INPUT SOURCES	1-10
1.4.1	Direct Command Format	1-10
1.5	Nesting Input Sources	1-11
1.6	Error Handling	1-12
1.7	Return and Abort Commands	1-13
1.8	KEYBOARD INPUT	1-15
1.8.1	Selecting commands in the menu window	1-15
1.8.2	Use of cursor control keys in option windows	1-18
1.8.3	Parameter entry in option windows	1-19
1.8.4	Function Keys	1-24
1.9	OBJECTS	1-26
1.10	SYMBOLIC DEBUGGING	1-26
2	MENU BASED COMMANDS	2-3
2.1	CONFIGURATION GROUP	2-4
2.1.1	Different Types of Memory	2-4
2.1.2	Pre-mapping and Block Mapping	2-5
2.2	DEBUGGER GROUP	2-10
2.2.1	Monitors	2-10
2.2.2	Hardware Monitors	2-10
2.2.3	Software Monitors	2-13
2.2.4	The Trace Facility	2-16
2.2.5	The Real Time Event Counter	2-18
2.3	TARGET GROUP	2-21
2.3.1	Running a Program	2-21
2.3.2	Analysis Commands	2-22
2.4	OBJECTS GROUP	2-24
2.4.1	Program Objects	2-24

2.4.2	Program Object Commands	2-24
2.4.3	The Trace Back Command	2-24
2.4.4	The Program Symbols Command	2-25
2.5	FILES GROUP	2-26
2.5.1	Editing A File	2-27
2.5.2	Displaying A File	2-28
2.5.3	Loading, Dumping and Patching a File	2-28
2.5.4	Command Files	2-30
2.5.5	System Commands	2-31
2.6	SERVICES GROUP	2-32
2.6.1	Session Logging	2-32
2.6.2	Window Managing Commands	2-33
2.6.3	Action Commands	2-34
3	DIRECT MODE COMMANDS	3-3
3.1	DIRECT INPUT MODE	3-3
3.2	COMMENT LINE	3-4
3.3	SHELL ESCAPE COMMAND	3-4
3.4	COMMAND HISTORY	3-6
3.5	EXPRESSIONS	3-7
3.5.1	Operands	3-7
3.5.2	Operators	3-8
3.5.3	Evaluation	3-10
4	COMMAND DIRECTORY	4-5
4.1	Overview	4-5
	action abort	4-7
	action end	4-8
	action if	4-9
	action pause	4-12
	action return	4-14
	action while	4-16
	border annotate	4-19
	border move	4-22
	clock modify	4-25
	clock show	4-28
	command all	4-29
	command command	4-32
	count disable	4-35
	count enable	4-36
	count modify	4-37
	count show	4-42
	dump all	4-45
	dump load	4-49
	edit all	4-53

edit command 4-57
edit source 4-60
hwmonitor disable 4-63
hwmonitor enable 4-66
hwmonitor Q-modify 4-69
hwmonitor show 4-76
hwmonitor T-modify 4-79
interrupt disable 4-87
interrupt enable 4-88
interrupt show 4-89
level cycle 4-91
level delete 4-92
level make 4-93
load all 4-95
load load 4-98
memory disasm 4-101
memory edit 4-104
memory fill 4-107
memory memdump 4-111
memory save 4-114
memorymap disable 4-117
memorymap enable 4-118
memorymap invert 4-119
memorymap modify 4-122
memorymap show 4-127
patch load 4-134
pod modify 4-137
pod show 4-139
processor halt 4-141
processor P-reset 4-142
processor run 4-144
processor wait 4-147
progsymb clear 4-149
quit 4-151
registers modify 4-153
registers show 4-156
session display 4-159
session modify 4-161
session resume 4-164
session suspend 4-165
swmonitor disable 4-167
swmonitor enable 4-169
swmonitor modify 4-171
swmonitor remove 4-175
swmonitor show 4-177
system chdir 4-179
system path 4-181

system remove 4-183
trace disable 4-187
trace enable 4-188
trace list 4-189
trace modify 4-191
trace show 4-195
variables describe 4-197
variables examine 4-199
view all 4-203
view command 4-206
view source 4-208
window cycle 4-211
window delete 4-212
window make 4-213
window state 4-215

5 DEBUGGING AN ASSEMBLY PROGRAM 5-3
5.1 Overview 5-3
5.2 System Requirements 5-3
5.3 Assembling the Sample Program 5-4
5.4 Invoking the Debugger 5-5
5.5 The Menu Window 5-6
5.6 Logging 5-9
5.7 Mapping the Emulation Memory 5-12
5.8 Loading and Running the Program 5-18
5.9 Examining the Program 5-21
5.10 Exiting from the Debugger and Patching 5-30
5.11 Program Listing 5-32

6 HARDWARE REFERENCE GUIDE 6-3
6.1 Introduction 6-3
6.2 Physical Dimensions 6-3
6.3 Environmental Conditions 6-3
6.4 Electrical Data 6-5
6.5 DC Specifications For Emulation Probe 6-6
6.6 Timing Specifications For Emulation Probe 6-10
6.6.1 Clock in MDS (12.5MHz) 6-10
6.6.2 Clock in Prototype (10MHz) 6-12
6.7 Extended Control Lines 6-16
APPENDIX A A-1
APPENDIX 2 ..... B-1

INDEX

CONVENTIONS USED IN THIS MANUAL

Enter	Indicates the *ENTER* key of the keyboard.
CTRL-x or ^x	Indicates that the *CTRL* key of the keyboard and the following key, in this case x, are to pressed simultaneously.
...	Indicates that the output on the screen is too wide to put all the displayed information onto the page of the manual. The user will, therefore, see additional information on his/her screen to that shown in the manual.
	All required input to the system is shown in blue.
	Italic characters are used throughout the manual for references to other manuals, filenames and commands used in text and markings on the hardware.

CONTENTS

		Page
1	DEBUGGING FACILITIES	1-3
1.1	STARTING A DEBUGGING SESSION	1-3
1.1.1	Invoking the Debugger	1-3
1.1.2	The Command File .debugrc	1-4
1.1.3	Altering the Parameters EDITOR and VIEW	1-5
1.2	HELP FACILITY	1-6
1.3	WINDOWS	1-7
1.3.1	Menu Window	1-8
1.3.2	Pop-up Windows	1-9
1.3.3	Display Windows	1-9
1.4	INPUT SOURCES	1-10
1.4.1	Direct Command Format	1-10
1.5	Nesting Input Sources	1-11
1.6	Error Handling	1-12
1.7	Return and Abort Commands	1-13
1.8	KEYBOARD INPUT	1-15
1.8.1	Selecting commands in the menu window	1-15
1.8.2	Use of cursor control keys in option windows	1-18
1.8.3	Parameter entry in option windows	1-19
1.8.4	Function Keys	1-24
1.9	OBJECTS	1-26
1.10	SYMBOLIC DEBUGGING	1-26

CHAPTER 1

DEBUGGING FACILITIES

This chapter explains some general debugging facilities, such as the different types of windows, terminal input, expressions and help information.

1.1 STARTING A DEBUGGING SESSION

1.1.1 Invoking the Debugger

The Debugger runs under Xenix (on PMDS-III) or Unix (PMDS-II) via a specially designed user interface, and is located in the directory */usr/bin* in the file *Adeb68010*. The following command invokes the Debugger:

```
Adeb68010
```

When the Debugger is invoked, the menu window is displayed on the top of the screen. The first message that appears on the display window is a release message, indicating the version number of the software and the debug unit and target microprocessor selected.

```
(C) 1987 Philips Export BV. All rights reserved.
PF8219M Debug processor asm-68010/68000   Release 2.0

 Selected debug unit: d0

 Connected MAB: PM8329/PM8328   Processor: 68000

 environment variable EDITOR defaulted to "vi"
 environment variable VIEW defaulted to "more"
```

A debugging session is terminated and control is returned to Xenix or Unix by selecting *quit* in the menu window.

1.1.2 The Command File .debugrc

When the Debugger is invoked, it searches for the command file (fully explained in Chapter 2) *.debugrc* in the current directory and in the user's home directory. If this file exists, the Debugger executes the commands in the file line by line and displays the results of each operation on the screen.

Executing the file *.debugrc* causes initialisation messages to disappear immediately, since the display window status is *overwrite* by default. If the first line of *.debugrc* changes the status of the display window to *append*, the messages will remain on the screen.

It should be noted that it is only advisable to put those

commands into the *.debugrc* file that are likely to be required for each debugging session. Otherwise, it will be necessary to either continually edit the *.debugrc* file or replace the commands and definitions in this file from within the Debugger.

eg The following OS (Xenix or Unix) command displays a command file *.debugrc* which opens a command log file, and maps emulation memory:

```
cat .debugrc

/ssm cmdlog=enabled cmdfile=logfile1
/cmm mode=user start=0 length=0xfff prot=rom
/cmm mode=user start=0x1000 length=0xfff prot=ram
/cmm mode=supervisor start=0 length=0xfff prot=ram
```

1.1.3 Altering the Parameters EDITOR and VIEW

Within a debugging session, two environment parameters are used: *EDITOR* and *VIEW*. The parameter *EDITOR* is set to the name of the editor used in the *Files Edit* command. The parameter *VIEW* is set to the name of the file pagination program used in the *Files View* command. By default, *EDITOR* is set to *vi*, and *VIEW* is set to *more*. The user may alter these default values before invoking the Debugger.

For example, when using the Bourne shell and the *emacs* editor, the following commands have to be issued before invoking the Debugger:

```
EDITOR=emacs
export EDITOR
```

When using the C shell instead, *EDITOR* has to be set as

follows:

```
set EDITOR emacs
```

1.2 HELP FACILITY

During a debugging session, help text can be displayed for all debugger commands and function keys.

Each help display consists of one or more pop-up windows. When the Help key is pressed, an overview of the keys which are used to display the various types of help information, is given:

Space Bar — When pressed, this key displays context-dependent help on the currently selected or preselected command.

+ and − — If more than one help window is available on a particular topic, the +/− keys may be used to page forward or backward through the help windows of that topic.

Function Keys — When a function key is pressed, help is shown on the function which the key performs in menu mode. When the Help key is pressed, a description of the use of function and cursor keys is given.

Esc — When pressed, this key terminates help processing and returns the Debugger to the state it was in before help was requested.

1.3.2 Pop-up Windows

A pop-up window shows command options (option window), help text (help window) or an error message (error window). A pop-up window is always surrounded by a double box.

Option Windows: many terminal commands have a set of options. When a terminal command with options is highlighted by the cursor box, the arrow down key or the initial letter of the command may be pressed to display the option window for this command. The user can now set parameters as required.

Help Window: The Debugger provides a help facility explained in Chapter 1. Help information is displayed in a pop-up window.

Error Window: Error messages and warnings are displayed in a pop-up window which can be deleted by pressing any key.

1.3.3 Display Windows

A display window is a window in which the output of executed commands is shown. One to four distinct display windows may be on the screen at any one time. Each of these display windows extends to the full width of the screen, but the number of lines may vary. In addition, one to four levels may be mapped on each window, giving 16 possible windows.

Display windows and levels may be created and deleted by means of function keys, described in Chapter 1, or menu commands, explained in Chapter 4.

Output is always directed to the default window. When the default window is set to auto-update, all its levels, including the invisible levels, are automatically updated.

1.4 INPUT SOURCES

The Debugger accepts command input from the keyboard, a command file, an action list or the history buffer. A command file is a file consisting of debugger commands. An action list is a sequence of debugger commands. The history buffer stores commands issued during a debugging session.

Keyboard input is described in Chapter 1, command files and action lists are described in Chapter 2, the history buffer is described in Chapter 3.

1.4.1 Direct Command Format

All commands in command files, action lists and the history buffer must be in direct command format. This means that each command must be accordance with the following syntax rules:

- A command name is specified by its full path through the menu.

- The full path of a command consists of a '/' referring to the top level of the menu, followed by the initial letter of the command group to which the command belongs, the initial letter of the command and the initial letter of the subcommand, if required.

- A command line may further contain one or more optional keyword assignments. By default, these option keywords are assigned their last used value. The pathname of a command and the specified options are separated by spaces. There is no fixed order in which options have to be entered on a command line.

1.3.2 Pop-up Windows

A pop-up window shows command options (option window), help text (help window) or an error message (error window). A pop-up window is always surrounded by a double box.

Option Windows: many terminal commands have a set of options. When a terminal command with options is highlighted by the cursor box, the arrow down key or the initial letter of the command may be pressed to display the option window for this command. The user can now set parameters as required.

Help Window: The Debugger provides a help facility explained in Chapter 1. Help information is displayed in a pop-up window.

Error Window: Error messages and warnings are displayed in a pop-up window which can be deleted by pressing any key.

1.3.3 Display Windows

A display window is a window in which the output of executed commands is shown. One to four distinct display windows may be on the screen at any one time. Each of these display windows extends to the full width of the screen, but the number of lines may vary. In addition, one to four levels may be mapped on each window, giving 16 possible windows.

Display windows and levels may be created and deleted by means of function keys, described in Chapter 1, or menu commands, explained in Chapter 4.

Output is always directed to the default window. When the default window is set to auto-update, all its levels, including the invisible levels, are automatically updated.

1.4 INPUT SOURCES

The Debugger accepts command input from the keyboard, a command file, an action list or the history buffer. A command file is a file consisting of debugger commands. An action list is a sequence of debugger commands. The history buffer stores commands issued during a debugging session.

Keyboard input is described in Chapter 1, command files and action lists are described in Chapter 2, the history buffer is described in Chapter 3.

1.4.1 Direct Command Format

All commands in command files, action lists and the history buffer must be in direct command format. This means that each command must be accordance with the following syntax rules:

- A command name is specified by its full path through the menu.

- The full path of a command consists of a '/' referring to the top level of the menu, followed by the initial letter of the command group to which the command belongs, the initial letter of the command and the initial letter of the subcommand, if required.

- A command line may further contain one or more optional keyword assignments. By default, these option keywords are assigned their last used value. The pathname of a command and the specified options are separated by spaces. There is no fixed order in which options have to be entered on a command line.

- An option consists of an option name, followed by '=' and a value. For each command, a list of option names which may be entered on the command call line is given in Chapter 4. Options which may take more than one value (see example below) must be entered as the last option on the command line.

eg `/fpa start=0 end=0x100 repl=0x12 0xcd 0x00 0x18`

'/' selects the top level of the menu. 'f' selects the *Files* command group. 'p' selects the *Patch* command. 'a' selects the *All* subcommand. The options 'start=0' and 'end=0x100' indicate that address range 0-0x100 has to be modified. 'repl=0x12 0xcd 0x00 0x18' means that the data sequence 0x12 0xcd 0x00 0x18 is repeatedly stored in the specified address range. As no file name is mentioned on the command line, the file name last used for the *Files All* subcommands is taken.

1.5 Nesting Input Sources

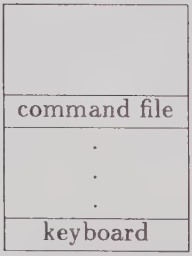
The Debugger maintains an input source stack, containing all input sources which have been invoked and not yet terminated.

On invoking the Debugger, input is taken from the keyboard. As a result, the keyboard is always on the bottom of the stack. When an input source is invoked during a debugging session, it is pushed on the stack. When execution of this input source is terminated, the input source is popped off the stack.

Input sources can be nested up to twenty levels.

eg When a command file invokes an action list, the action list is pushed on the top of the stack:

When execution of the action list terminates, it is popped off the stack and execution of the command file is resumed.



1.6 Error Handling

Two types of errors are distinguished: recoverable and unrecoverable ones. A wrongly spelled command name or option name always results in an unrecoverable error. Recoverable errors are always caused by illegal option values.

If a recoverable error is encountered while executing an input source, the option window which has erroneous values is

displayed. Fields which have been correctly entered are highlighted. Fields which are incorrect are red on colour monitors. The user may now correct option values. An /sar command can then be issued to return to the command file. If Esc is pressed, the error is treated by the debugger as unrecoverable.

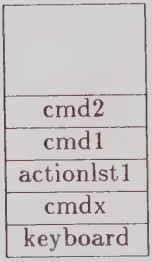
When an unrecoverable error is encountered, the input source that is currently executed is popped off the stack, and control is returned to the next item on the stack.

1.7 Return and Abort Commands

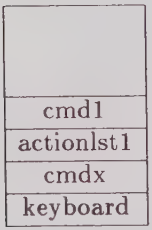
When a command file or action list contains a *Return* command, execution of the command file or action list is terminated, the input source is popped off the stack, and control is returned to the previous input source on the stack.

When a command file or action list contains an *Abort* command, execution of the command file or action list is terminated, and all input sources are popped off the stack until the keyboard is reached.

eg Assume that a command file *cmd1* invokes another command file *cmd2*.



When a *Return* command of *cmd2* is executed, control will be returned to *cmd1*.



When an *Abort* command of *cmd2* is executed, control will be returned to the keyboard.

1.8 KEYBOARD INPUT

Cursor keys and keys from the typewriter key area are used to select commands in the menu window and enter data in option windows.

Function keys are used to control display windows, invoke debugging facilities, and execute debugging commands.

1.8.1 Selecting commands in the menu window

Below is an overview of how menu commands can be entered, and how terminal keys can be used within the menu. Note that the way in which certain keys operate in the menu differs depending on the type of the command (terminal or intermediate) which is currently highlighted by the cursor box.

→ and ← — These keys move the cursor box from one command to another within the same level of the tree. When the cursor is located on the right-most command and → is pressed, the cursor is moved to the first command on the line. When the cursor is located on the left-most command and ← is pressed, the cursor is moved to the last command on the line. The command currently highlighted is called *preselected*. If it is an intermediate command, it may be selected for action by moving deeper in the tree. If it is a command terminator, it may be selected by pressing the Enter key. This causes the command to be executed.

↑ This key moves up one level in the tree. If it is pressed at the top level, no action is taken.

↓ If this key is pressed while the cursor box is located at an intermediate command, this command is selected. The lower line of commands in the tree replaces the top line of the menu. The cursor box is placed on the first command of the new top line, causing this to be the current *preselected* command.

If this key is pressed while the cursor box is located at a command terminator that has options, an option window is displayed. If the command terminator has no options, pressing the ↓ key is ignored.

Enter This key is ignored if pressed on an intermediate command. Otherwise, it causes the selection and execution of the command which is presently preselected.

Initial Letter If the initial letter of an intermediate command is pressed, then the indicated command is selected and a down arrow key is simulated. If the selected command is a command terminator, the command is just preselected and a down arrow key is simulated. This means that a pop-up window will appear in case of a command with options.

/ If the "/" key is pressed while a preselected command is on the menu, the present partially chosen command is aborted and the top level of the menu is reinstated.

Cmd Brk If this key is pressed while a command is being executed, that command will be terminated.

As indicated above, the user may select commands in a menu level by means of the arrow keys or the initial letter of a command. This is clarified by an example:

eg The *Debugger Count Show* command can be selected by using the arrow keys:

- press "/" to display the top level of the menu
- press the → key to highlight the intermediate command *Debugger*.
- press the ↓ key to select *Debugger*.
- press the → key twice to move the cursor to *Count*.
- press the ↓ key to select the *Show* command.

  The terminal command *Show* is preselected.

- pressing Enter displays the status and value of the real time counter.

eg The *Debugger Count Show* command can be selected by using its initial letters:

- press "/" to display the top level of the menu
- press the letter *d* to highlight the *Debugger* command
- press *c* to select *Count*

  The subcommand *Show* is now automatically

preselected, because *Show* is the first subcommand of *Count*.

- pressing Enter displays the status and value of the real time counter.

1.8.2 Use of cursor control keys in option windows

The following keys can be used within an option window except during text entry in a field.

↓	cursor down. When this key is pressed, the cursor is moved from one field to the field below.
↑	cursor up. When pressed, the cursor is moved from one field to the field above.
→	cursor right. When pressed, the cursor is moved from one field to the next field.
←	cursor left. When pressed, the cursor is moved from one field to the previous field.
Home	cursor home (first item of the window)
End	cursor end (last item of the window)
Esc	if an incorrect value is accidentally selected, press Esc to abort the option window, restoring the previous option values.
Enter	when the item you want to select is highlighted, press Enter to complete the selection.

when Enter is pressed twice, the option window is exited, the command is executed, and all options are set to the newly entered values.

Cmd Brk If this key is pressed while a command is being executed, that command will be terminated.

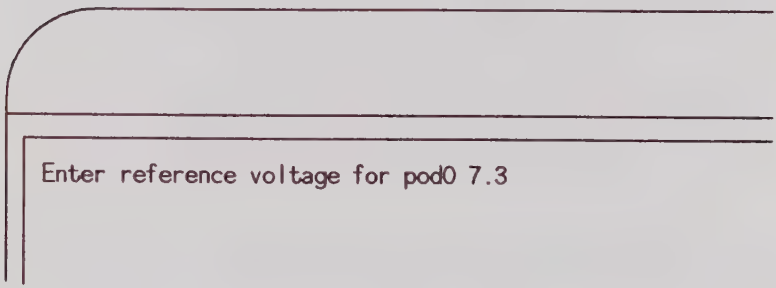
1.8.3 Parameter entry in option windows

Options are entered by using the character keys or by selecting them from a select-table range of choices. This is clarified by an example:

eg Select the option of the *pod* command:

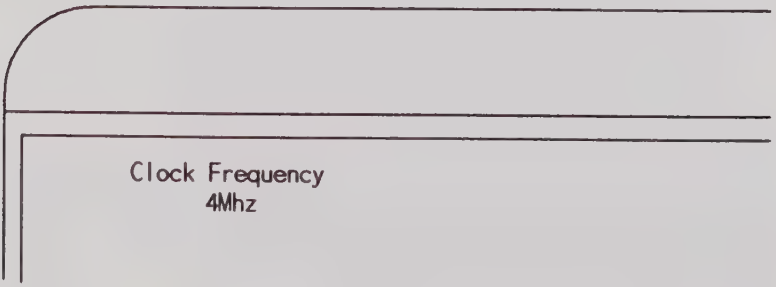
```
cpm

Enter reference voltage for pod0
```

Press the keys 7 , . and 3 to enter the value 7.3.

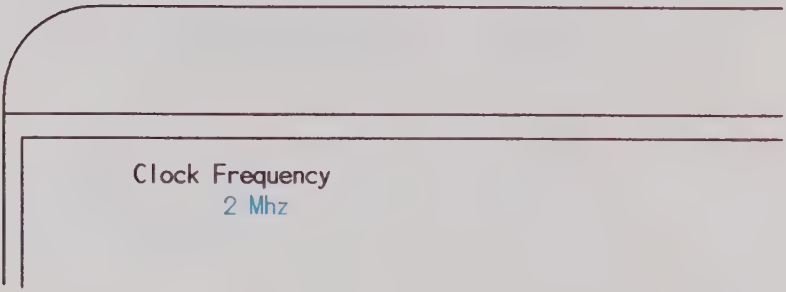


Press Enter twice to assign this voltage to pod0.

eg Select the option of the *Clock Modify* command.



Press the + key three times to select the frequency *2Mhz*.



Press Enter twice to assign this frequency to the clock.

When an option is entered by selecting it from a select-table range of choices, the following keys can be used:

+	the + key is the forward select key. When pressed, the next item in the select-table is displayed.
–	the – key is the backward select key. When pressed, the previous item in the select-table is displayed.

When an option is entered by editing text in a field, the following keys can be used. Text can be edited by either entering new options in the field or by pressing the Edit key to change the previously-entered options.

→	cursor right. When pressed, the cursor is moved one character to the right. Pressing cursor right when the cursor is on the last data character of the field is ignored.

←	cursor left. When pressed, the cursor is moved one character to the left. Pressing cursor left when the cursor is at the beginning of the line is ignored.
CTRL→	when pressing these keys, the cursor is moved to the word to the right of its current position. No characters are erased. Pressing this combination when the cursor is at the end of the field is ignored.
CTRL←	when pressing these keys, the cursor is moved to the word to the left of its current position. No characters are erased. Pressing this combination when the cursor is at the beginning of the field is ignored.
Home	the cursor is moved to the beginning of the field.
End	the cursor is moved to the end of the field.
CTRL **End**	when these keys are pressed, the character at the current cursor position and all following characters are deleted from the field.
Backspace	pressing this key erases the last character typed.
Enter	when you have entered an option value, press Enter to complete the selection.
Del	this key deletes the character at the current cursor position and slides all characters to its right, one position left. It is ignored when the cursor is at the end of the field.
Ins	this key toggles the input mode from auto-insert to over-strike.

Character Keys these keys may be used to enter values as shown in the examples above.

1.8.4 Function Keys

The function keys below are used to invoke debugging facilities. An overview of the corresponding keyboard keys is given in Appendix A.

Function Key Command	Function
Help	Invokes the Help facility
Dir Mode	Invokes direct input mode
Edit	Allows the user to edit the current field of an option window

The function keys below are used to execute debugging commands. An overview of the corresponding keyboard keys is given in Appendix A.

Function Key Command	Menu Comm.	Function
Step		Steps the user program from one emulation break to the next in sequence
Run	/tpr	Resumes program execution
Halt	/tph	Halts program execution
Cmd Brk		Terminate command, action list or command file

The function keys below are used to execute window managing commands. Corresponding keyboard keys are given in Appendix A.

Function Key Command	**Menu Comm.**	**Function**
Append	/sws	toggle *append to window* or *overwrite window* modes
MvBorder	/sbm	move the (top or bottom) border of the default window
Update	/sws	toggle the state of the default window between *alter* and *update*
Annotate	/sba	place a comment in the top border of the default window
Next Wnd	-	cycle the current window around the menu and display windows
Del Wnd	/swd	delete the default window
Make Wnd	/swm	create a new window
Cycle Wnd	/swc	cycle the default window around the display windows
Next Lvl	-	cycle the current window around the levels of the default window
Del Lvl	/sld	delete the displayed level of the default window
Make Lvl	/slm	create a new level of the default window
Cycle Lvl	/slc	cycle the default window around the levels of a display window

1.9 OBJECTS

Within the Debugger, *object* refers to a named entity used within the Debugger. Examples of objects are registers, command files, trace memory and section names defined by the user in his source program.

A special class of objects is *program objects.* These are symbols that are defined by the programmer while writing a source program.

1.10 SYMBOLIC DEBUGGING

The Debugger supports symbolic debugging. This means that, during a debugging session, both the user and the Debugger may use symbols, which are strings of characters, to represent objects, such as sections, labels, variables, files, registers, memory locations, etc. .

In order for the Debugger to be able to interpret source program symbols, the PCP Compilers and Assemblers create a record for each symbolic item and store them in a symbol table. The symbol table can then be loaded into the Debugger by the *Files Load* command which also loads the executable code into the target system. In general, the Debugger is able to interpret a symbol that is legal in the source program.

Note

Since assembler files generated from PCP C and Pascal compilers also contain symbolic information, limited high level language symbols are available for such files. Available are all symbols that are connected to absolute memory locations (for example, start

addresses of procedures/functions and addresses of static variables).

CONTENTS

		Page
2	MENU BASED COMMANDS	2-3
2.1	CONFIGURATION GROUP	2-4
2.1.1	Different Types of Memory	2-4
2.1.2	Pre-mapping and Block Mapping	2-5
2.2	DEBUGGER GROUP	2-10
2.2.1	Monitors	2-10
2.2.2	Hardware Monitors	2-10
2.2.3	Software Monitors	2-13
2.2.4	The Trace Facility	2-16
2.2.5	The Real Time Event Counter	2-18
2.3	TARGET GROUP	2-21
2.3.1	Running a Program	2-21
2.3.2	Analysis Commands	2-22
2.4	OBJECTS GROUP	2-24
2.4.1	Program Objects	2-24
2.4.2	Program Object Commands	2-24
2.4.3	The Trace Back Command	2-24
2.4.4	The Program Symbols Command	2-25
2.5	FILES GROUP	2-26
2.5.1	Editing A File	2-27
2.5.2	Displaying A File	2-28
2.5.3	Loading, Dumping and Patching a File	2-28
2.5.4	Command Files	2-30
2.5.5	System Commands	2-31
2.6	SERVICES GROUP	2-32
2.6.1	Session Logging	2-32
2.6.2	Window Managing Commands	2-33
2.6.3	Action Commands	2-34

CHAPTER 2

MENU BASED COMMANDS

This chapter explains the function of menu based commands. The menu based commands are divided into groups so that commands belonging to one group have similar functions or are frequently used together.

The command groups are explained in the following sections of this chapter. Each section begins with an overview of the function of the command group. Subsequently, a description of how to use commands of this group is given.

2.1 CONFIGURATION GROUP

The configuration group contains all those commands that are related to the configuration of the Debugger. A *Pod* command is provided to show and alter the operating voltage of the debugger pods. The clock speed of the target processor may be shown or selected by means of a *Clock* command. The *Interrupts* command controls the state of the target processor interrupts.

Before a program can be tested, memory can be emulated in which the program can be contained. Mapping is the process of determining what sort of memory, and how much memory has to be used, and where that memory is to be located. Mapping can be done by the *MemoryMap* command.

Target memory that is not yet available can be simulated by the emulation memory. This means that ROM memory areas of target memory are mapped into the RAM emulation memory with a ROM protection. Consequently, program development can be continued at the same time as hardware development without having to produce new PROM chips for each software change. With the target program located in RAM emulation memory, changes can be made when required. As program development continues, the debugged parts of a program can be transferred to target memory.

2.1.1 Different Types of Memory

Emulation memory contained in the AT-box is referred to as internal emulation memory, and is available in two types:

a) dynamic emulation memory of size 8, 16, 32, 64 or 256 Kbytes. With this memory, real-time emulation up to 8 MHz is possible.

b) static emulation memory of size 8, 16 or 64 Kbytes. With this memory, real-time emulation up to 8 Mhz is possible.

Optional emulation memory installed on the MAB can also be used. This memory allows emulation up to the maximum speed of the processor. It is referred to as external emulation memory.

All emulation memory is RAM. Nevertheless, memory areas with protection RAM or ROM will behave in an identical manner to their actual counterparts.

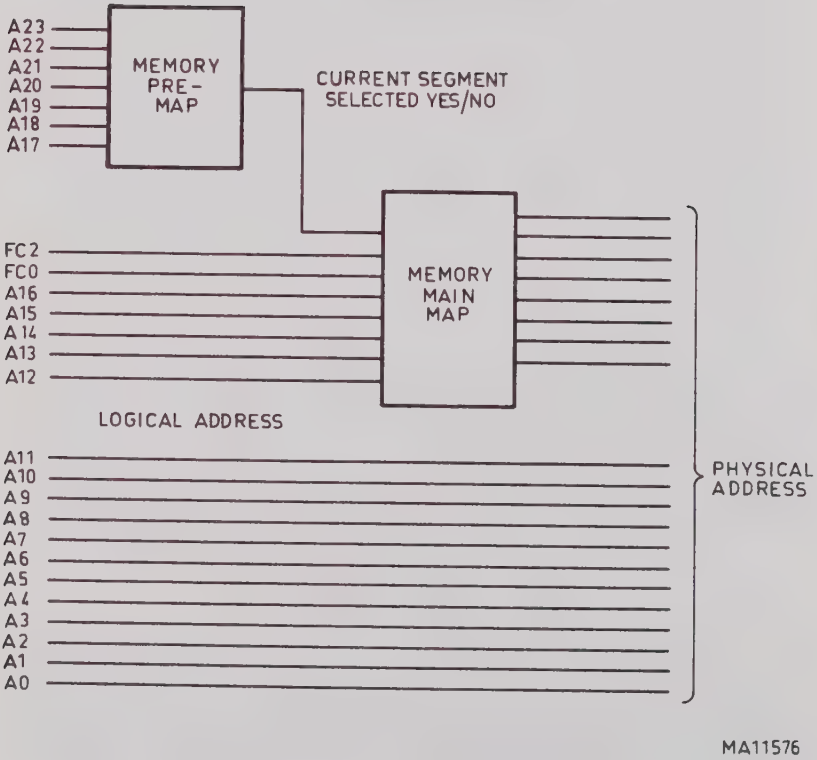
Note

Illegal memory access (for example, write access to a ROM-mapped area, or read/write of a guarded area) results in an Emulation Halt.

Furthermore, memory in the user's target system may be used.

2.1.2 Pre-mapping and Block Mapping

The 68000/68010 can address up to 16 Mbytes (24 address lines), whereas the Debugger map mechanism handles only 17 address lines. To enable the full address area to be handled, a *memory pre-mapping* technique is used. The diagram on the next page shows the relationship between logical and physical addresses.


A23
A22
A21
A20
A19
A18
A17
MEMORY PRE-MAP
CURRENT SEGMENT SELECTED YES/NO
FC2
FC0
A16
A15
A14
A13
A12
MEMORY MAIN MAP
LOGICAL ADDRESS
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
PHYSICAL ADDRESS
MA11576

Using the pre-mapping technique, the whole address range (16 Mbytes), for each mode (up, sp, ud, sd), is divided into 128 segments (determined by A17-A23). Each segment contains 128 Kbytes of memory (determined by A0-A16) and starts on a 128 Kbyte boundary. By using the *MemoryMap Modify* command, the user may select one of the 128 available segments to be the current segment, which is set by default to *target guarded*; the remaining 127 segments are all set to *target ram.* At the start of a debugging session, the currently active segment is 0 (addresses 0 - 0x1ffff).

eg To select the segment containing the address range 0x40000 to 0x5ffff, the following command is entered:

```
/cmm
```

```
Enter processor map parameters
 External mode:     up
 Start address:
       Length:
                              Emul Addr
        Class:     internal
   Protection:     ram
      Segment:     0x40000
```

The selected segment is the current segment for all modes (up, sp, ud, sd). The mode is determined by the lines FC0 and FC2. For each mode, the address range within the currently selected segment can be mapped to physical memory. Therefore a 128 Kbyte segment is divided into 32 (A12-A16) memory blocks

of 4 Kbytes (A0-A11) for all address modes. This technique is called block mapping.

The purpose of block mapping is to map blocks of the current segment to internal or external emulation memory or to target memory and to declare them as RAM, ROM or guarded (i.e. unused).

Within the current segment, the memory may be set using the *MemoryMap* command.

At the start of a debugging session, all memory within the current segment will be automatically defined as guarded. The minimum of memory which must be mapped to emulation or target memory is the memory space required to contain the microprocessor load-file. All remaining memory should be left guarded until actually called for in order to protect it from illegal accesses and program crashes.

eg An example of possible map settings is:

```
map : enabled
map segment = 0x000000 0x01ffff
map up  0x000000 0x001fff  =  internal  ram
map up  0x002000 0x01ffff  =  internal  rom
map ud  0x000000 0x001fff  =  external  ram
map ud  0x002000 0x01ffff  =  target    guarded
map sp  0x000000 0x01ffff  =  target    guarded
map sd  0x000000 0x01ffff  =  target    guarded
```

The *MemoryMap* command can also be used to display the map settings of the emulation memory, and the available or unavailable memory blocks. Furthermore, it is shown whether a processor address range is mapped to an emulation memory block.

2.2 DEBUGGER GROUP

When dealing with complex load modules, the user will find it necessary to define specific points in the program where a closer investigation is required. The commands in the *Debugger* group can be used to control and examine program execution.

2.2.1 Monitors

Monitors enable the user to predefine the circumstances (conditions) under which specific actions should be taken by the Debugger. A distinction between hardware and software monitors is made.

2.2.2 Hardware Monitors

These monitors are contained within the debugger hardware and consist of a condition and an action. The action specifies what has to be done when the condition is met. Both condition evaluation and actions are implemented in the debugger hardware and, therefore, real-time execution of programs is still possible when using hardware monitors.

Three different types of hardware monitors are provided:

1) *tr0* and *tr1* (trigger monitors)

2) *qr0* and *qr1* (qualifier monitors)

3) *buserror*, *clockfail*, *dma*, *haltline*, *notready*, *powerfail*, *rerun*, *resetline*

The hardware monitors *tr0*, *tr1*, *qr0* and *qr1* are part of the real-time control unit. The condition parts of these monitors include 16 or 24 address lines, 16 data lines, 4 control lines (eg read or write action, program reference, interrupt) and 8 or 0 test-pod lines. In addition, *tr0* and *tr1* have their own match counter which can count the number of times that the specified condition part is met; the match counter must have a value in the range 1-255 and it is set to 1 by default. The precise number of address and test-pod lines depends on the availability of the Break Trace Expansion unit. If this is installed, then 24 address lines may be specified but no test-pod lines. Otherwise, only 16 address and 8 test-pod lines may be specified.

The *HwMonitor Q-Modify* and *T-Modify* commands can be used to set up any or all of these condition parts.

The program address at which a specified condition is met, is called a *match point.*

When a hardware monitor of the first type (*tr0*, *tr1*) is enabled and its condition part is matched, one of the following actions is taken:

break The emulation run will be halted if no enabled software monitors (see below) are present which are triggered by the particular hardware monitor. If such software monitors are present and enabled, then the evaluation of these software monitors will be performed and the emulation run will not be halted (unless the software monitor itself causes it to be halted).

The break action generates a pulse action, explained below.

pulse A pulse will be generated, which can trigger either the event counter or an external device via the output sockets TR0 OUT and TR1 OUT on the MAB.

When a hardware monitor of the second type (*qr0*, *qr1*) is enabled and its condition part is matched, one of the following actions is taken:

break The emulation run will be halted if no enabled software monitors (see below) are present which are triggered by the particular hardware monitor. If such software monitors are present and enabled, then the evaluation of these software monitors will be performed and the emulation run will not be halted (unless the software monitor itself causes it to be halted).

pulse An internal pulse will be generated for the real-time trace mechanism. This pulse is necessary for qualified tracing.

All hardware monitors of the third type have a fixed condition and a fixed action. The fixed action is *break*, which means that the emulation run is halted. The condition for each of these monitors is fully described in Chapter 4, *HwMonitor*.

Both the *Enable* and *Disable* commands allow the user to enable or disable a monitor of the first or third type. When a hardware monitor is disabled, no action is taken. The hardware monitors of the second type (*qr0* and *qr1*) cannot be disabled. However, when their action is set to *pulse* and they are not used for qualified tracing, the action has no effect.

2.2.3 Software Monitors

In contrast to hardware monitoring where both condition evaluation and actions are performed in real-time, software monitoring requires software intervention for performing these functions. As a consequence, programs will no longer be executed in real-time.

A software monitor consists of a trigger, a condition and an action list. The status of the monitor is either enabled or disabled. The trigger defines the type of emulation break for which the action list is executed. The action list is only executed when the monitor is enabled and the condition is met.

At the beginning of a debugging session, no software monitors are defined. During the session, the user can define up to 20 software monitors. When defining a software monitor, the user specifies the monitor name which must be unique.

The following triggers can be specified:

instr — the monitor is activated each time an assembly instruction is about to be executed.

tr0, tr1, qr0 or qr1 — the monitor is activated each time that the conditions specified by a hardware monitor (for which the action is *break*), have been met.

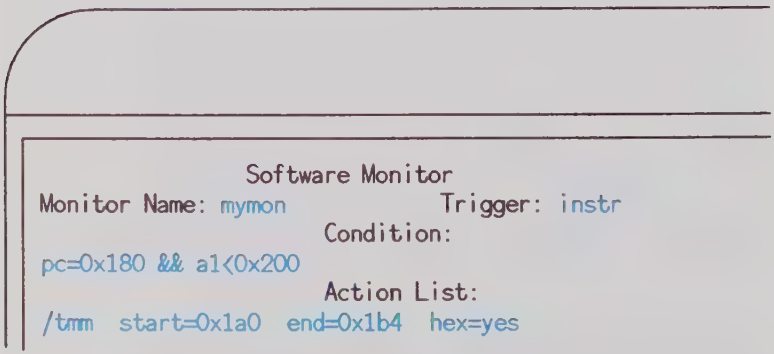
rtec — the monitor is activated when a *break* is generated by the real-time event counting mechanism.

As mentioned above, the action list specifies the commands to be executed when the condition is met, the software monitor is

enabled and activated by the trigger.

Note that the condition is always satisfied when the Condition field is set to 1.

eg /dsm



A software monitor *mymon* is created. Each time an instruction has been executed, the Debugger checks whether the program counter has the value 0x180 and the register *a1* has a value smaller than 0x200. If this is the case, the contents of the memory locations 0x1a0 - 0x1b4 are displayed in hexadecimal format.

As trigger is set to *instr*, the Debugger checks after each instruction whether the program counter has the value 0x180. This checking is performed in software and, therefore, it may take a long time before the point is reached where the program counter contains the value 0x180. The following method which uses the hardware monitor qr0 as a trigger, will be faster:

/dhq

```
                    Qr. Hardware Monitors

Hwm
                    Condition           Addr Mode

qr0  Addr  0x180                        all
     Data
     Pod                                Action:
     Ctrl                               Break

qr1  Addr                               all
     Data
     Pod                                Action:
     Ctrl                               Pulse
```

```
/dsm
```

```
                    Software Monitor
Monitor Name: mymon              Trigger: qr0
              Condition:
a1<0x200
              Action List:
/tmm  start=0x1a0  end=0x1b4  hex=yes
```

The hardware monitor *qr0* is used to generate a *break* when

program execution has reached address 0x180. So the program runs in real-time until the specified address is reached. Then the value of a1 is checked. If it is smaller than 0x200, the action list of swmonitor *mymon* is executed.

2.2.4 The Trace Facility

The real-time trace memory is totally separate from the emulation memory in the system. Thus, during an emulation run, there will be no down-grading or conflict of either memory.

The trace memory contains 256 entries of 48 bits, in which microprocessor bus information can be stored while executing a program. When program execution is halted, the user can obtain information about what happened prior to program termination by examining the contents of the trace memory.

Data can be entered unconditionally into trace memory, which means that all microprocessor information is entered in trace memory during an emulation run. It is also possible to enter data into trace memory only when the conditions defined by one or both hardware monitors *qr0* and *qr1* are met. As such, the data that is entered into trace memory can be qualified by the contents of the address or data bus, control line status, test pod condition, or any combination of these conditions. Note that these conditions can be specified through the *HwMonitor* commands.

The clock source which will be used to provide triggering for the trace sampling logic can be machine cycles, either of two test pod signal transitions, or program references.

The user can specify the location of the trace window around the match point. The entries of the real-time trace memory can

record the conditions existing on the bus in three different ways:

pre-tracing — on detection of a match point, the emulation halt will stop the tracing. At that moment the *real-time trace memory* contains up to 255 entries preceding the match point.

centre-tracing — the emulation run, on detection of a match point, will continue making trace memory entries for a further 127 steps. This means that the real-time trace memory will contain the last (up to 127) entries before the match point and the first (up to 127) entries after the match point.

post-tracing — the emulation run, after detection of a match point, will continue until a further 255 trace entries are recorded. The real-time trace memory will thus contain the first entries (up to 255) after the match point was detected.

During an emulation run, the trace memory entries are continually refreshed on a first-in/first-out basis ensuring that an up-to-date record of the emulation run is maintained. When the emulation run has stopped, the contents of the real-time trace memory can be examined and shown in disassembled format wherever possible.

eg The contents of the real-time trace memory entries −40 to −35 are shown, after issuing a *Trace List* command:

```
/dtl
```

The options Start Line and End Line are set to −40 and −35 respectively:

```
Show Real Time Trace Memory
Start Line: -40
End Line: -35
```

When pressing Enter twice, the command is executed and the following output is displayed:

```
-40  up  000086  0838  btst     #0x1,0x1ffc    mr/pr
-39  up  000088  0001                          mr/pr
-38  up  00008a  1ffc                          mr/pr
-37  up  00008c  67f8  beq.s    0x86           mr/pr
-36  ud  001ffc  ff02                          mr
-35  up  00008e  11d9  move.b   (a1)+,0x1fff   mr/pr
```

2.2.5 The Real Time Event Counter

The Real Time Event Counter is a facility with which events such as absolute time, program execution cycles, machine cycles, program references, interrupt acknowledges, high to low transitions of pin 6 of pod 0, and tr0 and tr1 condition matches, can be counted.

It is possible to specify that an emulation break must occur when the counter reaches a certain value.

The hardware monitors *tr0* and *tr1* can be used to define conditional starts and stops of the event counting mechanism by selecting the *from* and/or *to* options in the *Count Modify* command.

eg When the condition specified by the hardware monitor *tr0* is met, the counting mechanism starts counting the absolute time in micro-seconds.

/dcm

```
Modify Counter Settings
Counter rtec:
Event:  mc
Value:
 From:  tr0
   To:  No trigger
```

When pressing Enter twice, the following output is displayed:

```
counter rtec : enabled
actual value of counter rtec = 0
counter rtec : enabled
counter rtec = mc 1 from tr0
```

2.3 TARGET GROUP

The target group consists of target control commands and commands frequently executed with these target control commands.

2.3.1 Running a Program

After issuing a *Run* command, program execution is started. The program then runs until it encounters an emulation break, at which point debugging information can be displayed on the screen. If the program goes into an infinite loop, it can be terminated by entering the *Halt* command.

The *new* option is provided for the *Run* command. If this option is set to *Yes*, program execution begins at the address determined by the option *default value*. In addition, the trace memory is cleared.

An option is present to specify the number of steps after which program execution must be halted. The step size is determined by the emulation breaks that occur. An execution step is completed each time an execution point is reached where at least one action list of a software monitor is executed.

Note that not all commands can be executed while the target program is running. If an unacceptable command is entered when the program is running, the system will respond with the error message:

Processor is running

A *Wait* command can be used to inhibit the execution of any further commands until the emulation run is halted or until a

certain amount of time has expired.

Once program execution is halted, the target processor can be reset by issuing a *P-Reset* command. As a result, the registers and status of the processor may have been altered. By default, the target is reset using the processor reset vector which is located at addresses 0 to 7 in *sp* mode. An option allows the user to reset the target processor without using the processor reset vector.

2.3.2 Analysis Commands

When the execution of a program has been halted, the user may review the contents of the registers, display a program disassembly, or show and affect memory.

The contents of the registers of the target processor can be displayed and altered by the *Registers* command.

The *Memory* commands are used for inspecting or changing memory locations.

Two commands are used to display memory locations: *Memory Memdump* and *Memory Disasm*. The *Memory Memdump* command displays an area of memory in octal, decimal, ASCII or hexadecimal format. A program disassembly of a specified memory range can be displayed by using a *Memory Disasm* command. As with all debugger commands, symbolic references can be used to identify memory locations.

Changing memory locations is sometimes useful when the program fails to produce the intended results. The program can then be rerun to see whether the changes work. Two commands are used to modify memory locations: *Memory Edit* and *Memory Fill*. The *Memory Edit* command is used to display an area of

memory. After making the display window the new current window, the user may edit this memory area. The *Memory Fill* command is used to set the contents of specified memory locations to specified values.

Any changes made during debugging affect only the program in the emulation memory or in the target RAM. Source files can be changed with the aid of the text editor, after which the programs must be reassembled.

It is easy to keep a record of temporary changes of the modules of memory locations, since the contents of these locations can be saved in a file by means of a *Memory Save* command. This file will be in *out.h* type format and will not contain any debug information or symbol tables.

A program disassembly of a specified memory range can be displayed. As with all debugger commands, symbolic references can be used to identify memory locations.

2.4 OBJECTS GROUP

2.4.1 Program Objects

The commands in the program group reference program objects, which are symbols defined by the programmer in his source program.

2.4.2 Program Object Commands

The value of program objects can be displayed by issuing an *Examine* command. By means of the *Describe* command, information about a program object can be displayed.

2.4.3 The Trace Back Command

When program execution is halted, the *Tracebk* command may be used to display the current call stack, i.e. the execution path followed to reach the current location.

For each function the provided information consists of:

- actual values of the parameters

When the option *'Do you want local stack variables?'* is set to *Yes*, the actual values of the local variables are also displayed.

2.4.4 The Program Symbols Command

When a file is loaded in a debugging session, symbolic information is loaded by default in a debugger table. To reduce overhead, this symbolic information may be deleted later in the session by means of a *Program Symbols Clear* command.

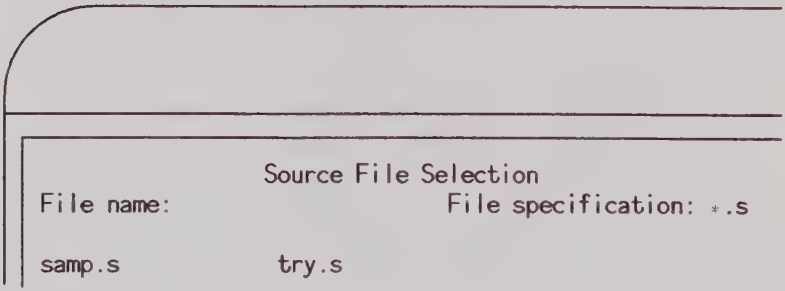
2.5 FILES GROUP

This group contains all the commands that are related to file handling. The commands allow the user to edit, display and delete files. Load files in *out.h* format can be loaded in the Debugger. Command files, explained further on in the section, may be executed.

As mentioned in the *PCP Cross-Assembler 68000 Manual*, some file types are determined by the suffix to their file name. All assembly source files are suffixed *.s*. It is convenient to suffix command files with the sequence *.cmd*. The user could end all load files with the same file extension, say *.out*. All *Files* commands have subcommands, each of them recognising a file type by means of a wild card pattern. The default wild card pattern is *.s* for source files, *.cmd* for command files, and * (indicating that any file type may be specified) for *Load* and *All* files. The user may alter the wild card pattern of a subcommand. This can be done by altering the setting of the *File specification* option. For instance, when all load files are suffixed *.out*, it is useful to set the file specification of all *Load* subcommands to *.out*.

A default file name is kept across all the *Files* subcommands recognising a particular file type. For instance, when the source file *samp.s* is referenced by the *View Source* file, the default file name of the *Edit Source* command is *samp.s*.

The option window of a subcommand handling a particular file type shows all the files of this file type that exist in the current working directory. The file specification is displayed at the top right-hand corner of the window. For instance, the option window of a *Source* subcommand shows all the files ending with the *.s* extension which are present in the current working directory.



The user can select one of these files by using the arrow keys. Files not residing in the current working directory can be selected by typing their name.

All files, no matter what their file type, can be selected by means of the *All* subcommand.

2.5.1 Editing A File

The *Files Edit* command allows the user to edit a file within the Debugger environment. Unless the user has set the keyword parameter *EDITOR* to a particular OS (Xenix or Unix) editor before invoking the Debugger, the *vi* screen editor will be used to edit files. For more information, see Chapter 1.

A file cannot be edited within a window on the screen. This is because most screen editors consider that they have the whole screen for their exclusive use, which is incompatible with the Debugger's screen usage. Therefore, when executing a *Files Edit* command, the menu and display windows disappear from the screen. The entire screen is then available to edit the specified file. After editing the file, the user may press any key to restore the information shown in the windows before executing the *Files Edit* command.

2.5.2 Displaying A File

A *Files View* command produces a display of the specified file in the default window. Unless the user has set the keyword parameter *view* to a particular OS (Xenix or Unix) pagination program before invoking the Debugger, the *more* file viewer will be used to view files. For more information, see Chapter 1.

When issuing a *Files View* command, the menu and display windows disappear from the screen. The entire screen is then used to display the specified file. The file pagination program *more*, explained in the *Xenix Command Reference Manual* (Xenix) or the *User Manual, Volume II* (Unix), may now be used to view different parts of the file.

After viewing the file, the user may press any key to restore the information shown in the windows before executing the *Files View* command.

2.5.3 Loading, Dumping and Patching a File

When the locations of various memory modules have been defined, the *Files Load* command is used to read the load file containing the user's object code program from the disk, into the assigned areas of memory. By default, the symbol table will be loaded into the Debugger at the same time. The code is verified each time it is written to memory.

When loading a file, four options can be specified. The option *Mode* specifies the mode in which the load file has to be loaded. The option *Use symbolic info* can be used to suppress the loading of the symbol table in the Debugger's table. This is particularly

useful to reduce loading overhead when no symbolic references are required. The option *Download to target* allows suppression of the loading of the object code into memory. Furthermore, the option *Verify memory writes* specifies whether or not the loaded data is to be verified each time object code is written to memory.

eg The file *myfile* is loaded.

```
fla

                    Load a File
              Mode : user        Download to target : Yes
Use symbolic info : Yes       Verify memory writes : Yes

 File name: myfile     File specification: *

cc.cmd          samp          a.out     myfile
```

The *Dump* command can be used to dump a load file. This means that the contents of the load file can be displayed in octal, hexadecimal, decimal or ascii format. It is also possible to display instructions of the load file in disassembled form.

The *Patch* command provides the capability of modifying a load file.

2.5.4 Command Files

Commands can be entered into a file which can then be called from within the Debugger by issuing a *Files Command* command. The Debugger will execute the commands in the file line by line. This type of file is referred to as a command file. Command files may be executed as many times as required and, as such, they are particularly useful for executing frequently-used command sequences.

A command file can be created by whichever editor is available on your system. All commands must be entered on a separate line in direct command format, which is explained in Chapter 1.

All empty lines and comment lines in a command file will be ignored when executing the file. Comment lines consist of a # character at the first position of the line, followed by an arbitrary string of characters. Shell commands in a command file will not be executed.

The Debugger exits from a command file after having executed the last command or a *Return* or *Abort* command or if the Cmd Brk key is pressed.

One command file can be invoked from another command file by the use of a *Files Command* command.

Note: option values in a command file may not have more characters than the width provided for the option value on the equivalent screen. If the option value is too long, a syntax error message will be shown.

2.5.5 System Commands

The system commands allow the user to enter some OS (Unix or Xenix) commands during a debugging session. The current working directory may be changed by the *Chdir* command. A *Remove* command is provided which allows the user to delete a chosen file.

The *Path* command can be used to specify the default search path for the directory containing a file. The value of path is a string containing one or more directory names. Each alternative directory is separated by a colon (:). The initial path is ".".

When a file required is not present in the current working directory, the user must either add the directory path of the file to the *Path* option or use the full path name, if possible.

Note that *Path* is not applicable to the *Files Edit* and *Files View* commands. For these commands, only the current working directory is searched for a specified file.

It is possible to compile source files on one machine (for example, VAX, PMDS-II) and debug on another machine (for example, PMDS-III). In this case, the object file and all source files compiled must be copied to the machine on which the Debugger is installed.

2.6 SERVICES GROUP

The *Services* command group consists of session commands, window managing commands and action commands. Session commands permit the user to create and subsequently close command and display log files. Action commands are concerned with the execution of action lists. Window managing commands are commands related to the control of basic display windows on the screen. Each of these commands is explained below.

2.6.1 Session Logging

During a debugging session, two sorts of information can be recorded in two separate files. First, commands issued can be logged in a command log file. Secondly, command output can be logged in a display log file.

Log files can be opened (enabled) and closed (disabled) by issuing a *Session Modify* command. This command also allows for selective control and change of the current log files. When specifying two different file names, the command input and the debugger's output to commands executed will be logged in two separate files. When only one file name is specified, both command input and output will be logged in this one file. At any one time, only one command log file and one display log file can be open.

All issued commands, including those that contain errors, are recorded in direct command format. Commands containing errors are logged as comment lines. Therefore, command log files or edited versions thereof can be executed as a command file in a debugging session by means of the *Files Command* command. Only the modified fields of a command are logged.

Command output is recorded in a display log file. Error messages are also recorded. Except for a # character at the beginning of each response line, the command output is identical to what you see in the display window(s) on the screen. Lines beginning with a # character are treated as comment lines. Consequently, when command input and output are logged in the same file, this file can also be executed as a command file in a debugging session.

The *Session Suspend* and *Session Resume* commands suspend and resume logging to both the command and display log files.

The *Session Show* command shows the current log files.

2.6.2 Window Managing Commands

Window commands are provided to cycle the default window around display windows, to create and delete display windows. It is also possible to set the way debugger commands are added to the default display window. There are two possible modes: *append to window* and *overwrite window*. Moreover, an *Auto-Update* facility is provided: when setting *Auto-Update* to *Yes* by means of the *Update* key or the *Services Window State* command, the default display window is updated each time an emulation run is halted.

Level commands are provided to cycle the default window around the levels of a display window, to create and delete a level of the default window.

Border commands are provided to re-arrange the display area of each display window, and to display a command line in the top border of a display window.

In the right-hand top corner of each display window, a window

status box is shown. This box contains the following information:

- path of the command last executed in the window.

- window number (W1, W2, W3 or W4).

- level number (L1, L2, L3 or L4).

- mode, which is either Overwrite (Ov) or Append (Ap).

- the characters Au when Auto-update is set to Yes.

eg The window status box below shows that the relevant display window is window 2 level 3. The command last executed in the window is /dhq. The window mode is currently set to Overwrite.

```
/DHQ W2 L3 Ov
```

The only time when window commands are not available is when pop-up windows are on the screen or the screen updating is disabled (such as when a non well-behaved shell command is active). When pop-up windows are on the screen, the basic display windows may be covered, and therefore cannot be changed.

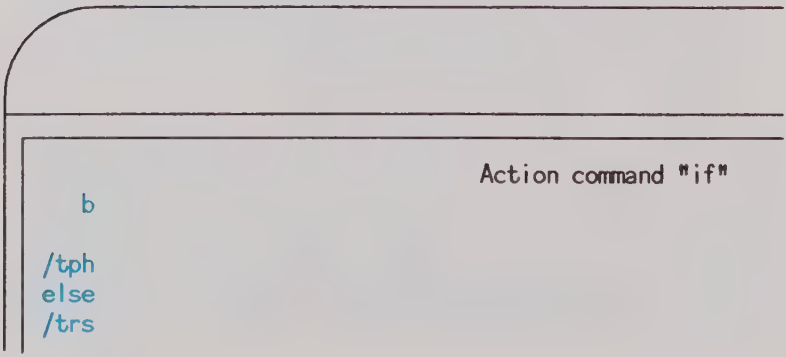
2.6.3 Action Commands

The Debugger normally executes all commands in a sequential order. The *If* and *While* commands can be used to alter the sequence in which instructions are read from a command stream and subsequently executed. They are particularly useful in software monitor action lists and command files.

The *If* and *While* commands execute a range of debugger commands in direct command format, called an action list, when a specified condition is met. The condition is an expression.

The *If* command consists of a condition, an action list and an optional *else* part. The condition is evaluated and when it has a non-zero value (true), the action list will be executed. If it has a zero value (false) and an *else* part is specified, the action list specified by the *else* part will be executed instead. Note that the user has to type the keyword *else* when an *else* part needs to be specified; no *then* keyword needs to be specified.

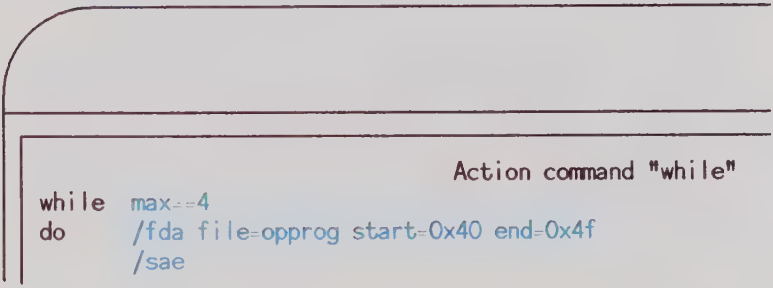
eg /sai



The expression b is evaluated. If it has a non-zero value (true), the emulation run will be halted. If *b* has a zero value (false), the values of all registers are displayed. *b* is the condition.

The *While* command consists of a *while* part specifying a condition and a *do* part specifying an action list. The condition is repeatedly evaluated and as long as it has a non-zero value (true), the action list is executed.

eg /saw



Note that the /sae (*Services Action End*) command is optional in an action list entered in an option window. When it is not specified by the user, the /sae command is implicitly assumed by the Debugger.

When an action list contains a *Return* command, execution of the action list is stopped, and control is returned to the stream that invoked the command file. This stream can be an action list, a command file, or the terminal. Note that when the *Return* command is used together with an *If* or *While* command, it enables the user to decide whether to continue the execution of the command file. The Cmd Brk key may be used to abort an action list.

CONTENTS

		Page
3	DIRECT MODE COMMANDS	3-3
3.1	DIRECT INPUT MODE	3-3
3.2	COMMENT LINE	3-4
3.3	SHELL ESCAPE COMMAND	3-4
3.4	COMMAND HISTORY	3-6
3.5	EXPRESSIONS	3-7
3.5.1	Operands	3-7
3.5.2	Operators	3-8
3.5.3	Evaluation	3-10

CHAPTER 3

DIRECT MODE COMMANDS

Direct mode commands are commands that are not apparent to the user because they are internal to the Debugger or implicit for ease of access.

3.1 DIRECT INPUT MODE

As described in Chapter 1, commands can be selected from the menu window using cursor or character keys. All commands executed are entered in a history buffer and a log file, if enabled.

The mode in which commands are selected from the menu window is called menu mode. Certain functions cannot be used in this mode because there is a conflict between the use of the cursor keys for these functions and the menu mode. All these functions can be used in direct input mode.

In direct input mode, comment and shell escape commands can be executed, expressions can be evaluated, and the history buffer can be accessed. Direct input mode can also be used to enter the value of an option when the actual command has been selected in menu mode. This is particularly useful when the space provided in the option window is insufficient to enter the option value.

Note that it is not possible to enter menu commands in direct mode.

The user can switch from menu mode to direct input mode by pressing the [Dir Mode] key. In direct input mode, the menu window is no longer displayed on the screen. The first line of the screen now contains an empty line, surrounded by a double box.

The user may return to menu mode by pressing [Esc] .

3.2 COMMENT LINE

The comment command allows the user to issue comment lines in the log output file and the history buffer. A comment can be entered in direct input mode only. The "#" character must be in the first position of the comment line, and may be followed by a free format string. This string has a maximum length of 256 characters.

eg # This is a comment line.

In the log output file and the history buffer, a comment string appears exactly as the user has typed it.

3.3 SHELL ESCAPE COMMAND

It is possible to enter an OS (Xenix or Unix) command while working in direct input mode during a debugging session. The command must be preceded by an exclamation mark (!), but is otherwise input in the normal manner. When executing the command, the screen is cleared and subsequently used for standard output. Once the command is completed, the information shown on the windows before executing the shell command is restored and the user is returned to menu mode.

eg 1 !date

displays the current system date and time. Any key may now be pressed to return to menu mode.

eg 2 As an example of executing OS (Xenix or Unix) commands from within the Debugger, a new shell is started through the *sh* command:

!sh

It will then be possible to perform such tasks as updating source files and recompiling, all without terminating the current debugging session. To terminate the new shell and return to the current debugging session, simply enter:

CTRL-d

Note that when a source file has been updated and recompiled in this way, it will be necessary to return to the current debugging session, delete the currently loaded *load file* and then load the updated version of the same file.

Refer to the *Xenix Command Reference Manual* (Xenix) or the *User Manual, Volume II* (Unix) for a more detailed description of how to use these commands.

While running a shell escape command, no other debugging commands can be entered.

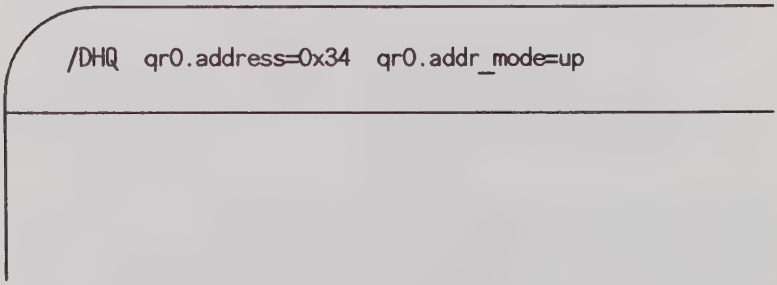
Shell commands and their output are not logged and they do not appear in the history buffer.

3.4 COMMAND HISTORY

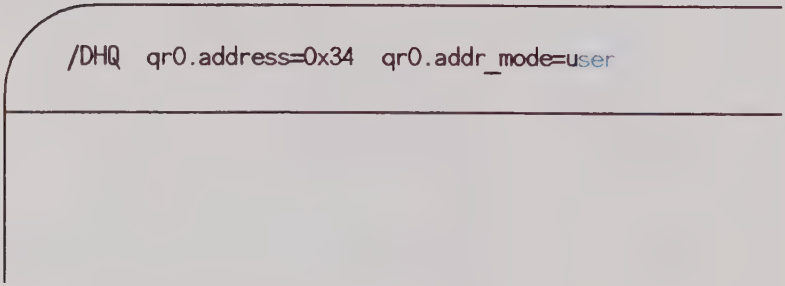
Commands issued during a debugging session are entered in direct command format in an internal history buffer, which stores up to twenty commands. When working in direct mode, the history mechanism can be used to re-enter these commands.

The command history is accessed via the cursor up and down keys. When pressing the cursor up key, the currently typed command line is abandoned and the most recently entered command in the history buffer is displayed on the top line of the screen. The user can then press cursor up or down keys to select other history lines. The cursor left, cursor right, Home , and End keys may be used to move the cursor on a history line. The backspace key may be used to edit this line. The currently selected command can be executed by pressing Enter . When pressing Esc the history line mode is exited.

eg Press |Dir Mode| to work in direct input name. After pressing ↑ once, the last issued command is displayed on the screen:



Press End to locate the cursor at the end of the line. Use the backspace key to replace the character 'p' by the characters 'ser'.



Pressing Enter executes the command.

3.5 EXPRESSIONS

Expressions are composed of operands and operators. Parentheses may be used to alter the order of evaluation of compound expressions.

All expressions may be entered in direct input mode. This can be done by pressing Dir Mode , entering the required expression, and pressing Enter . The output appears in the default display window.

It is possible to enter more than one expression on the same line. In this case, commas and blanks must be used as separators.

3.5.1 Operands

Operands may be numbers, character constants and references to objects.

Numbers

Decimal, octal, hexadecimal and binary number representations are supported by the Debugger; a floating point representation is not yet available. By default, decimal numbers are used. Hexadecimal, octal and binary numbers are preceded by 0x (or 0X), 0o (or 0O), and 0b (or 0B), respectively.

eg	371	is a decimal number
	0xa20	is a hexadecimal number
	0o25	is an octal number
	0b1011	is a binary number

Character Constants

Character constants are enclosed in single quotes. A backslash is used as an escape character to introduce certain special characters:

newline	'\n'
horizontal tab	'\t'
backspace	'\b'
carriage return	'\r'
form feed	'\f'
backslash	'\\'
single quote	'\''
bit pattern	'\ddd'

where ddd is one to three octal digits, specifying the value of the required character.

3.5.2 Operators

The following monadic and dyadic operators are supported:

Monadic Operators

+	unary plus
−	unary minus
!	logical negation
~	one's complement

The logical negation operator ! may not occur as the first character of a line, as a ! at the beginning of a line is regarded as an escape to the operating system.

Dyadic Operators

+	addition
−	substraction
*	multiplication
/	integer division
%	remainder
<<	left shift
>>	right shift
<	smaller than
>	greater than
<=	less than or equal to
>=	greater than or equal to
!=	not equal to
==	equal to
&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
&&	logical AND
||	logical OR

3.5.3 Evaluation

Expression evaluation is performed using signed long representation on the host machine. Consequently, expressions evaluated by the Debugger may not always yield the same result as a similar operation encoded in a program and executed on the target system.

CONTENTS

		Page
4	COMMAND DIRECTORY	4-5
4.1	Overview.	4-5
	action abort	4-7
	action end	4-8
	action if	4-9
	action pause	4-12
	action return	4-14
	action while	4-16
	border annotate	4-19
	border move	4-22
	clock modify	4-25
	clock show	4-28
	command all	4-29
	command command	4-32
	count disable	4-35
	count enable	4-36
	count modify	4-37
	count show	4-42
	dump all	4-45
	dump load	4-49
	edit all	4-53
	edit command	4-57
	edit source	4-60
	hwmonitor disable	4-63
	hwmonitor enable	4-66
	hwmonitor Q-modify	4-69
	hwmonitor show	4-76
	hwmonitor T-modify	4-79
	interrupt disable	4-87
	interrupt enable	4-88
	interrupt show	4-89
	level cycle	4-91
	level delete	4-92
	level make	4-93
	load all	4-95
	load load	4-98

memory disasm 4-101
memory edit 4-104
memory fill 4-107
memory memdump 4-111
memory save 4-114
memorymap disable 4-117
memorymap enable 4-118
memorymap invert 4-119
memorymap modify 4-122
memorymap show 4-127
patch all 4-131
patch load 4-134
pod modify 4-137
pod show 4-139
processor halt 4-141
processor P-reset 4-142
processor run 4-144
processor wait 4-147
progsymb clear 4-149
quit 4-151
registers modify 4-153
registers show 4-156
session display 4-159
session modify 4-161
session resume 4-164
session suspend 4-165
swmonitor disable 4-167
swmonitor enable 4-169
swmonitor modify 4-171
swmonitor remove 4-175
swmonitor show 4-177
system chdir 4-179
system path 4-181
system remove 4-183
trace disable 4-187
trace enable 4-188
trace list 4-189
trace modify 4-191
trace show 4-195
variables describe 4-197
variables examine 4-199
view all 4-203
view command 4-206
view source 4-208

window cycle 4-211
window delete 4-212
window make 4-213
window state 4-215

CHAPTER 4

COMMAND DIRECTORY

4.1 Overview.

This chapter contains an alphabetical list of commands. The following information is supplied for each command:

Path: contains the command nodes which have to be selected in order to execute the command.

Function Keys: keys which have to be pressed in order to execute the command (where applicable).

Option Window: shows the option window, containing all command options in **menu mode** (where applicable).

Option names: lists all option names in **direct command format** (where applicable).

Description: explains the command. Options and their values are explained the way they are used in menu mode. Nevertheless, all option values can equally be used in direct command format.

Examples: one or more examples of how to use the command in menu mode and in direct command format.

PATH:

/
Services
Action
Abort

DESCRIPTION:

This command aborts all command streams except for the basic command stream which becomes the current command stream.

EXAMPLES:

1) When an error occurs during the execution of command files, a new terminal command stream is opened. If the user then decides to restart the execution of all command files, he may enter an ***abort*** command. As a result, all active command streams are terminated and the Debugger returns to the basic command stream:

```
/saa
Enter
```

PATH:

/
Services
Action
End

DESCRIPTION:

This command marks the end of an *If* or *While* or software monitor action list. When specifying an action list interactively, the *Action End* command is automatically included.
When specifying an action list in a command file the */sae* is obligatory.

EXAMPLES:

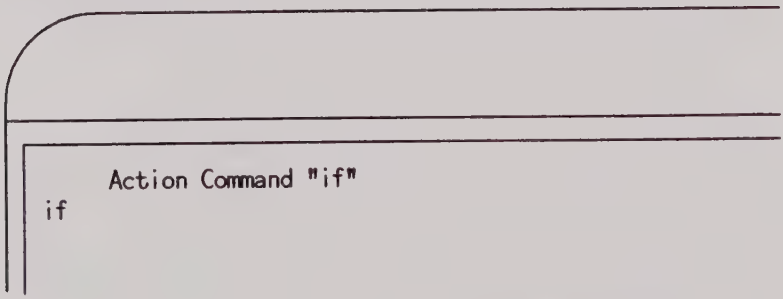
1) Assume that an action list of a *While* command is being executed. Issuing the following command terminates execution of this action list:

   /sae
   Enter

PATH:

/
Services
Action
If

OPTION WINDOW:



OPTION NAME (direct command format):

cond

DESCRIPTION:

The *if* command consists of an *if* part specifying a condition, a *then* and an optional *else* part both specifying action lists. During an emulation run the debug software will test the specified condition. If the condition evaluates to true, the action list specified by the *then* part will be executed. If the condition evaluates to false, the action list specified by the *else* part, if any, will be executed.

The maximum number of commands allowed in an *if* action list, is 25. This number includes the commands of the *else* part, but

excludes the *else* line.

It is possible to nest *if* commands.

EXAMPLES:

1) The following command specifies that the value of all registers has to be displayed when the value of the program counter equals 800 (hexadecimal). Otherwise, the value of the label *l1* has to be displayed.

```
sai
```

```
     Action Command "if"
if pc==0x800

/trs
/sae
else
/ove names=l1
```

Enter Enter

2) In direct command format, the previous command has to be entered as follows:

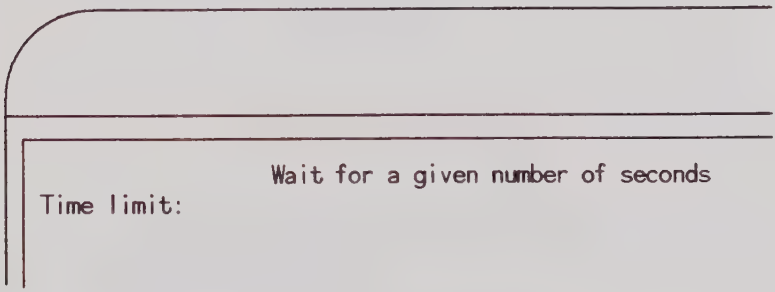
```
/sai cond=pc==0x800
     /trs
```

```
/sae
 else
/ove names -l1
```

PATH:

/
Services
Action
Pause

OPTION WINDOW:



OPTION NAMES (direct command format):

time

DESCRIPTION:

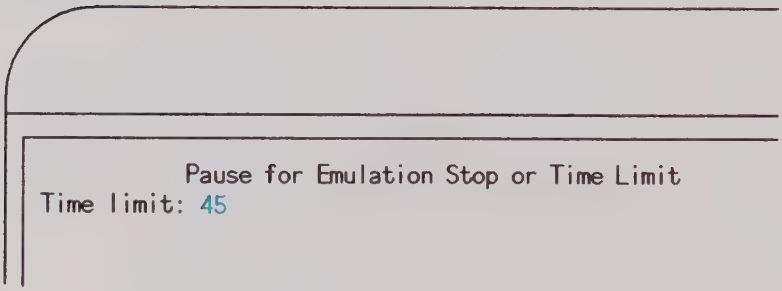
After issuing a *pause* command, no further user commands are executed until the time limit is expired.

The default action of this command is to execute the previous *pause* command. The option for this command allows specification of the time limit in seconds.

EXAMPLES:

1) After issuing the following command during an emulation run, no user commands will be executed for the first 45 seconds. As a result, the output displayed on the screen can be examined for at least 45 seconds.

```
'sap
```



Enter Enter

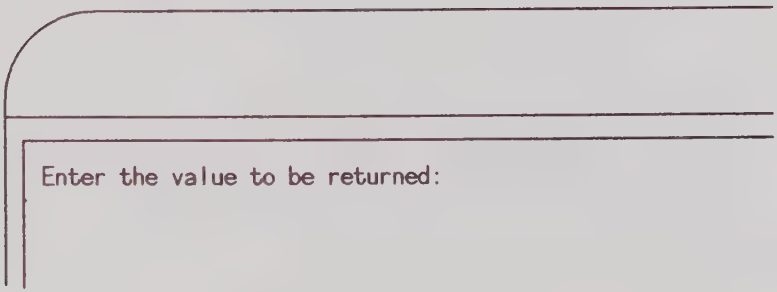
2) In direct command format the previous command is entered as follows:

```
/sap time=45
```

PATH:

/
Services
Action
Return

OPTION WINDOW:



OPTION NAMES (direct command format):

retval

DESCRIPTION:

The *return* command halts execution of a non-basic command stream.

The option allows specification of the integer return code, which is the value to be returned. By default, this value is zero.

EXAMPLES:

1) Execution of a command file will be stopped when the following command is encountered:

```
/sar
Enter Enter
```

2) Execution of a command file will be stopped when the following command is encountered. The value returned is 1.

```
/sar
Enter the value to be returned: 1
Enter Enter
```

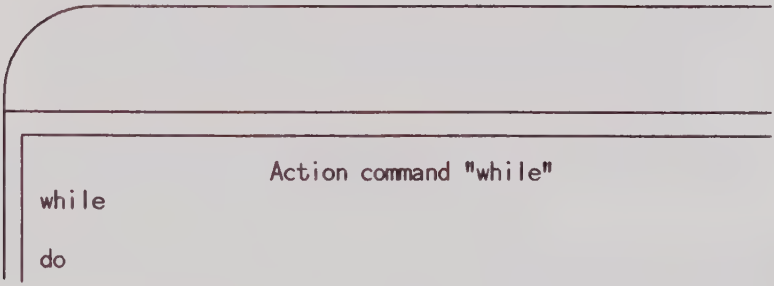
3) In direct command format, the previous command is entered as follows:

```
/sar retval=1
```

PATH:

/
Services
Action
While

OPTION WINDOW:



OPTION NAMES (direct command format):

cond

DESCRIPTION:

The *while* command consists of a *while* part specifying a condition, and a *do* part specifying an action list.

The *while* command executes the list of commands, specified by the action list, as long as the specified condition evaluates to a non-zero value (true). The list of commands must be entered in direct command format.

The maximum number of commands allowed in a *while* action list, is 25.

EXAMPLES:

1) As long as the value of the program counter is greater than 100 (hexadecimal), all registers and the symbol *a* are shown.

```
'saw
                                  Action command "while"
        while                     pc>0x100

        do                        /trs
                                  /ove names=a
```

Enter Enter

2) In direct command format the previous command may be entered as follows:

```
/saw cond=pc>0x100
    /trs
    /ove names=a
    /sae
```

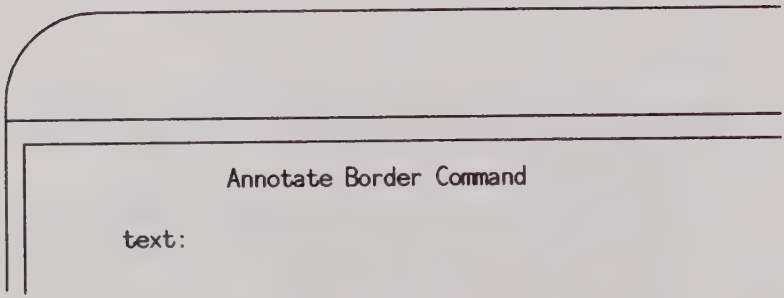
PATH:

/
Services
Border
Annotate

FUNCTION KEYS:

Annotate

OPTION WINDOW:



OPTION NAMES (direct command format):

text

DESCRIPTION:

The user may place a comment up to 39 characters in the top border of the display window. This can be done by either issuing a /sba command or pressing the function key Annotate.

- After issuing the command /sba, the option Text may be set to the required comment. After pressing [Enter] twice, the comment appears in the comment field in the window border.

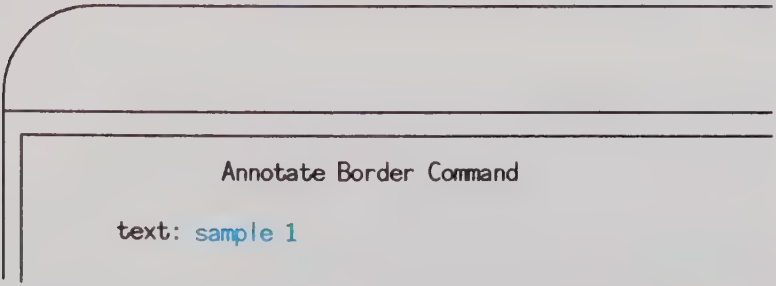
- After pressing the function key [Annotate] , the cursor is moved into the comment field in the window border where a free format string can be typed. Once the user presses [Enter] , the comment remains in the window border. When pressing [Esc] instead, the comment is aborted.

The comment is associated with the display level of the window. This means that when a new level is created or selected, the comment and the corresponding old level will be preserved.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and one display window.

   After entering /sba a comment, say *sample 1*, may be typed.



   After pressing [Enter] twice, the comment appears in the comment field in the window border.

2) Assume that the basic screen currently consists of the menu window and one display window.

   After pressing Annotate a comment can be entered. For example, the comment *sample 2* may be typed.

3) In direct command format, the previous command is entered as follows:

```
/sba text=sample 2
```

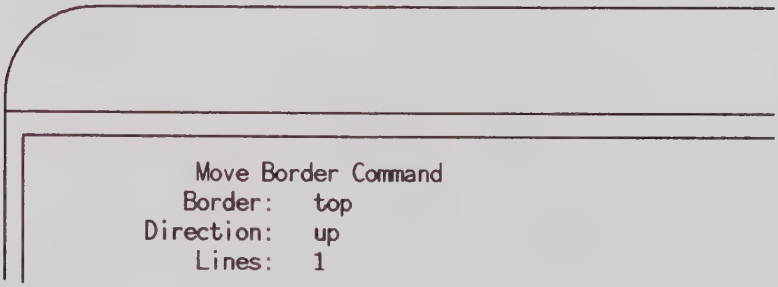
PATH:

/
Services
Border
Move

FUNCTION KEYS:

MvBorder

OPTION WINDOW:



OPTION NAMES (direct command format):

border
direction
lines

DESCRIPTION:

The display area of a display window may be re-arranged. This can be done by means of a menu command or a function key.

When using the menu command /sbm, the following options have to be specified:

Border Specifies whether the top or bottom border of a display window has to be moved.

Direction Specifies whether the border has to be moved up or down.

Lines Specifies the number of lines that the border has to be moved in the specified direction.

When using the function key, first make the window of which the borders are to be moved the current window. Subsequently, press the key [MvBorder] . Pressing either the cursor up or cursor down key then identifies the top or bottom border to be moved. Once the user has chosen the border to be moved, further cursor movements up or down move the selected border position up or down. Not until the [Enter] key is pressed, is the border actually moved. The visual indication of where the border is finally positioned is indicated by the current cursor position.

The border of a display window cannot be moved when it is adjoining the menu window or the bottom of the screen, or when it would remove another display window from the screen. Note that each display window consists of a minimum of two screen lines.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and two display windows. Window 2 is the default window.

To move the top border of Window 2 two lines upwards, press [MvBorder] once and ↑ three times. Pressing [Enter] actually moves the border.

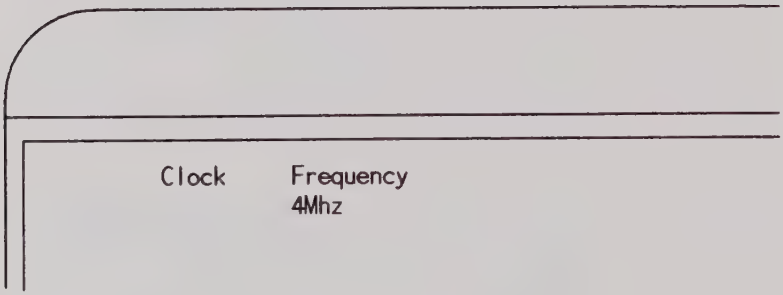
2) In direct command format, the previous command is entered as follows:

```
sbm lines=2
```

PATH:

/
Config
Clock
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

freq

DESCRIPTION:

The *clock* command allows the clock speed of the target processor to be selected.

The clock frequency may be selected from a select-table range of frequencies which have the following meanings:

2 Mhz When a frequency of 2 Mhz is specified, the clock is delivered by the debugger hardware.

4 Mhz When a frequency of 4 Mhz is specified, the clock is delivered by the debugger hardware.

8 Mhz When a frequency of 8 Mhz is specified, the clock is delivered by the debugger hardware.

12.5 Mhz When a frequency of 12.5 Mhz is specified, the clock is delivered by the debugger hardware.

target When target is specified, the target clock will be used.
Note: this option can only be used when the emulation probe is plugged into the target system.

external When this argument is selected, the clock must be delivered from an external source, such as a pulse generator. The clock will be applied to the system via the TTL CLOCK-IN BNC connector on the MAB.

By default the clock is set to 4 Mhz.

EXAMPLES:

1) The clock is set to 12.5 MHz.

```
ccm

Clock       Frequency
            12.5Mhz
```

Enter Enter

```
clock = "12.5"
Microprocessor reset because of clock change.
```

2) In direct command format, the previous command is entered as follows:

```
/ccm freq=12.5
```

PATH:

/
Config
Clock
Show

DESCRIPTION:

The *clock show* command displays the clock speed of the target processor.

EXAMPLES:

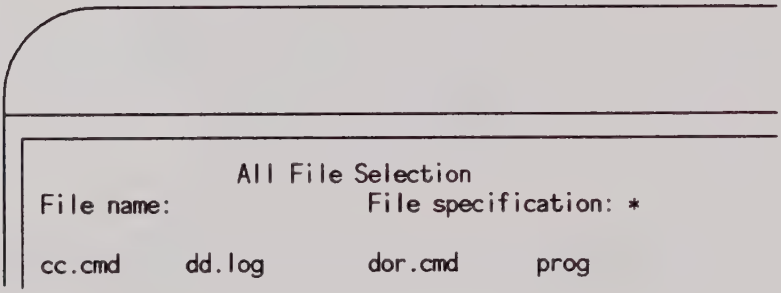
1) The current value of the clock speed of the target processor is shown:

```
/cc
Enter

clock = "12.5"
```

PATH:

/
Files
Command
All

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

This command executes any command file. The command file is executed line by line until the last command, a *Return*, an *Abort* or an *End* command is encountered.

By default, the current default file for the *Files All* group is executed. The option for this command allows the user to specify a file name.
The first time the command is used, no default file name is present.

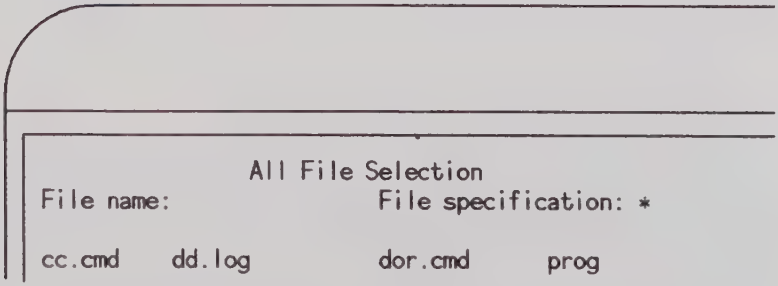
File name Specifies the name of the file to be edited. The file name can be selected from the list of displayed files which are present in the current working directory. This can be done by using the arrow keys to highlight a file name and by subsequently pressing [Enter] . The user can also type the command file name using the character keys.

File specification Specifies that all file names will be shown in the menu. The user may alter the file specification.

EXAMPLES:

1) The following command allows the user to select a file which has to be executed:

```
'/fca
```



The file *cc.cmd* is executed when pressing ↓ [Enter] [Enter] .

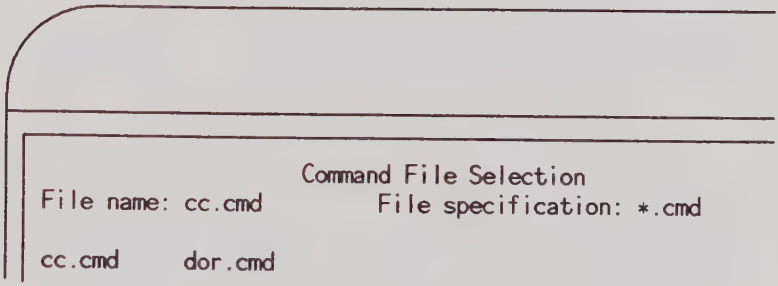
2) In direct command format, the previous command is entered as follows:

```
/fca file=cc.cmd
```

PATH:

/
Files
Command
Command

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

This command shows all files ending with the suffix *.cmd*. The command file *cc.cmd* is executed line by line until the last command, a *Return*, an *Abort* or an *End* command is encountered.

By default, the current default file for the *Files Command* group is executed. The option for this command allows the user to specify a file name.
The first time the command is used, no default file name is present.

File name Specifies the name of the command file to be edited. The file name can be selected from the list of displayed files, which are the files present in the current working directory and ending *.cmd*. This can be done by using the arrow keys to highlight a file name and by subsequently pressing Enter . The user can also type the command file name using the character keys.

File specification Specifies that all file names, ending *.cmd* will be shown in the menu. The user may set the file specification to a different wild card match.

EXAMPLES:

1) The following command executes the current default file for the command files group.

```
/fc
Enter
```

2) In direct command format, the previous command may be entered as follows:

```
/fcc file=cc.cmd
```

PATH:

/
Debugger
Count
Disable

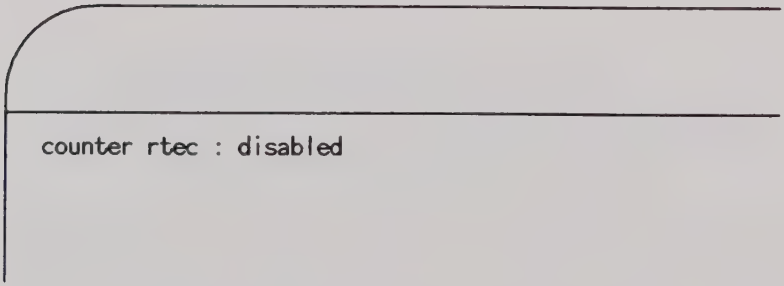
DESCRIPTION:

This command disables the counting mechanism. At invokation of the debugger, the counter is disabled.

EXAMPLES:

1) The counting mechanism is disabled:

/dcd
Enter



PATH:

/
Debugger
Count
Enable

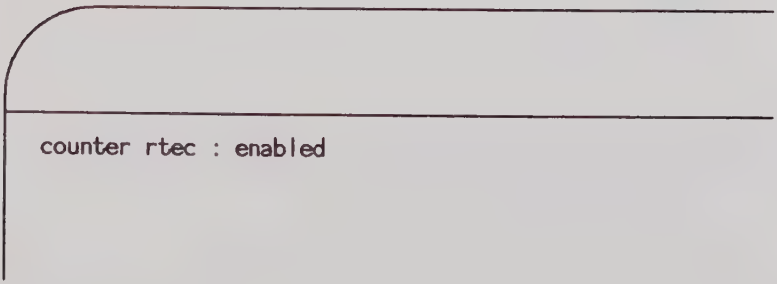
DESCRIPTION:

This command enables the counting mechanism.

EXAMPLES:

1) The counting mechanism is enabled:

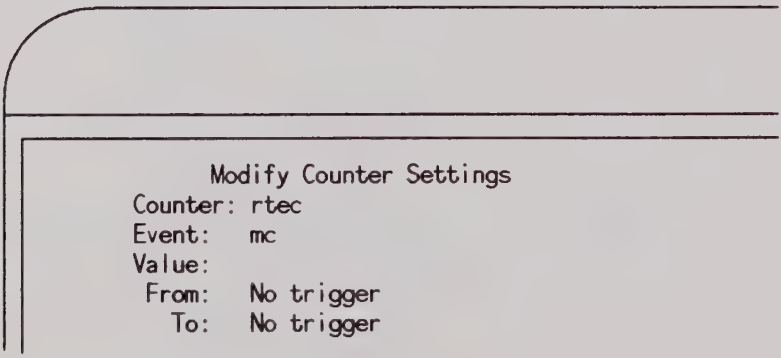
```
/dce
Enter
```



PATH:

/
Debugger
Count
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

event	value
from	to

DESCRIPTION:

This command changes the current counter settings, and enables the counter. The options have the following meanings:

Event Specifies the event of which occurrences have to be counted. The + or − keys may be used to specify one of the following events:

us	absolute time (in micro-seconds).
cc	microprocessor clock cycles.
pref	user program or supervisor program accesses (program references).
mc	machine cycles. This is the default event.
tr0	the contents of the target system bus are as specified by the hardware monitor *tr0*.
tr1	the contents of the target system bus are as specified by the hardware monitor *tr1*.
pod06hl	high to low transitions of pin 6 of pod 0.
ina	interrupt acknowledges.

Value Specifies that an emulation break must be generated when the count reaches the specified number.

The specified expression must have a value in the range 1 to 8388605 (decimal) when representing micro-seconds, and in the range 1 to 16777210 (decimal) otherwise.

From Specifies the start condition of the counting mechanism. This condition can be one of *tr0*, *tr1* and *no trigger* (*No_trigger* in direct command format). By default, the start condition is set to *no trigger* which means that the count starts at the beginning of the emulation run. The + or − keys can be used to set the start condition to *tr0* or *tr1*. In this case the count starts where the condition specified by the selected hardware monitor is met.

To Specifies the end condition of the counting mechanism. This condition can be one of *tr0*, *tr1* and *no trigger* (*No_trigger* in direct command format). By default, the end condition is set to *no trigger* which means that the count stops at the end of the emulation run. The + or − key can be used to set the end condition to *tr0* or *tr1*. In this case the count stops when the condition specified by the selected hardware monitor is met.

EXAMPLES:

1) The following command specifies that the microprocessor clock cycles have to be counted after the condition specified by tr0 is met and until the condition specified by tr1 is also met:

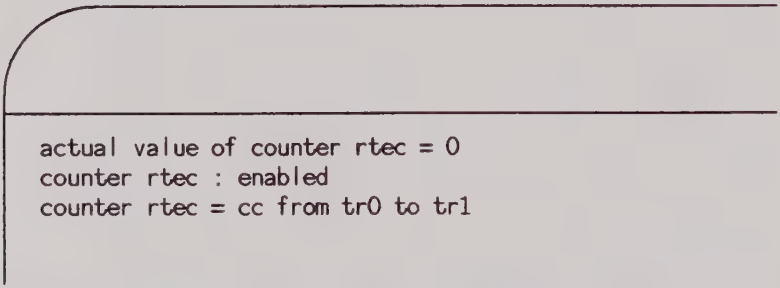
```
/dcm
```

```
Modify Counter Settings
Counter: rtec
Event:   mc
Value:
 From:   No trigger
   To:   No trigger
```

Use either of the keys + or − to set event to *cc*. Subsequently, use ↓ and − to set From to *tr0* and To to *tr1*.

```
Modify Counter Settings
Counter: rtec
  Event:   cc
  Value:
   From:   tr0
     To:   tr1
```

Enter Enter



```
actual value of counter rtec = 0
counter rtec : enabled
counter rtec = cc from tr0 to tr1
```

2) In direct command format, the previous command may be entered as follows:

```
/dcm event=cc from=tr0 to=tr1
```

PATH:

/
Debugger
Count
Show

DESCRIPTION:

This command displays the current settings of the real time event counter. This means that the following information is given:

- the actual value of the counter
- the state of the counter, which is either enabled or disabled
- the event of which occurrences have to be counted, which may be one of *us*, *cc*, *pref*, *mc*, *tr0*, *tr1*, *pod0hl*, *ina*. A description of each of these events is given in the *Count Modify* command.

EXAMPLES:

1) The settings of the real-time event counter are displayed:

```
/dcs
```

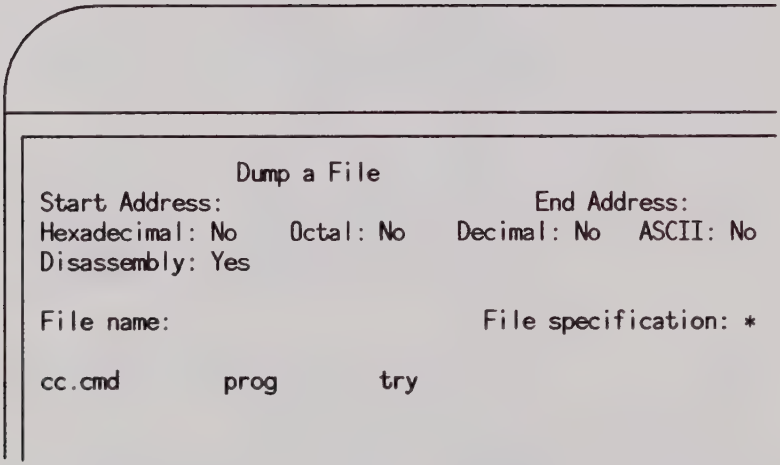
```
actual value of counter rtec = 2
Counter rtec : enabled
Counter rtec = mc from qr0
```

The value of the counter is 2. The counter is enabled and it will count the number of microprocessor clock cycles during an emulation run. The counter starts at the first match of qr0 and stops at the end of the emulation run.

PATH:

/
Files
Dump
All

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
hex	octal
decimal	ascii
disa	file
spec	

DESCRIPTION:

This command displays the contents of a load file in the required format.

Start Address Specifies the first location that has to be dumped.

End Address Specifies that the address range beginning at *Start Address* and ending at *End Address* has to be dumped. If this option is not specified, only the *Start Address* will be dumped.

Hexadecimal When this option is set to its default value *No*, then the load file is not displayed in hexadecimal format. The + or − keys can be used to set the option to *Yes*, in which case the load file is displayed in hexadecimal format.

Octal Specifies whether or not the load file is to be displayed in octal format.

Decimal Specifies whether or not the load file is to be displayed in decimal format.

ASCII Specifies whether or not the load file is to be displayed in ASCII format.

Disassembly Specifies whether or not all the instructions of the specified address range are to be displayed in disassembled form. This is the default format.

Note that a load file can be displayed in several formats at any

one time. At least one format should be set to *Yes* to have output on the screen.

Specifying *disassembly* will disable all other output radices.

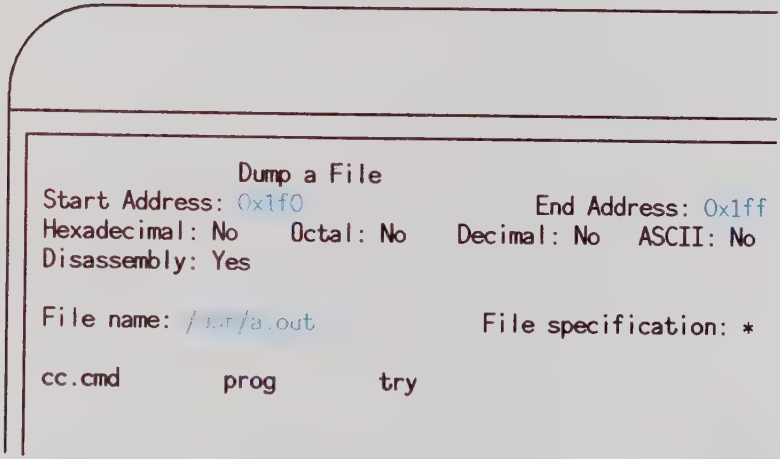
File name Specifies the name of the load file to be dumped. The file can be selected from the list of displayed files. This is done by first selecting the option *File name*, and secondly pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required file. The user can also type the load file name using the character keys.

File specification Specifies that all file names will be shown in the menu. The user may alter the file specification.

EXAMPLES:

1) The contents of the address range 0x1f0 - 0x1ff of the load file */usr/a.out* is displayed in disassembly format when entering:

```
/fda
```



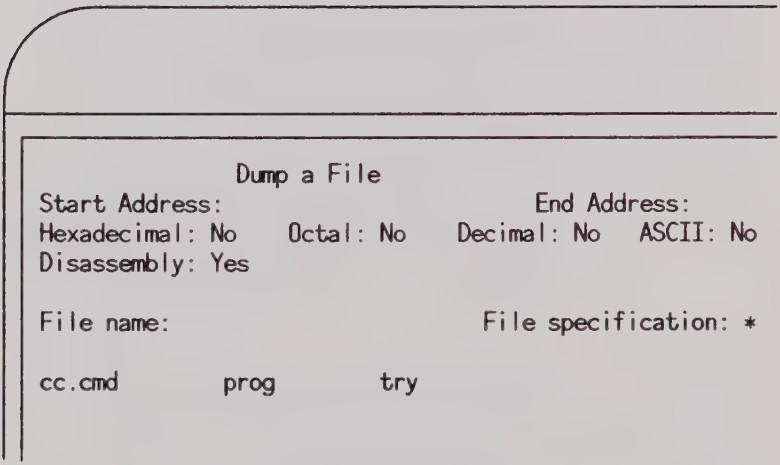
2) In direct command format, the previous command may be entered as follows:

```
/fda start=0x1f0 end=0x1ff file=/usr/a.out
```

PATH:

/
Files
Dump
Load

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
hex	octal
decimal	ascii
disa	file

DESCRIPTION:

This command displays the contents of a load file in the required format.

Start Address Specifies the first location that has to be dumped.

End Address Specifies that the address range beginning at *Start Address* and ending at *End Address* has to be dumped. If this option is not specified, only the *Start Address* will be dumped.

Hexadecimal When this option is set to its default value *No*, then the load file is displayed in hexadecimal format. The + or − keys can be used to set the option to *Yes*, in which case the load file is displayed in hexadecimal format.

Octal Specifies whether or not the load file is to be displayed in octal format.

Decimal Specifies whether or not the load file is to be displayed in decimal format.

ASCII Specifies whether or not the load file is to be displayed in ASCII format.

Disassembly Specifies whether or not all the instructions of the specified address range are to be displayed in disassembled form. This is the default format.

Note that a load file can be displayed in several formats at any

one time. At least one format should be set to *Yes* to have output on the screen.

Specifying *disassembly* will disable all other output radices.

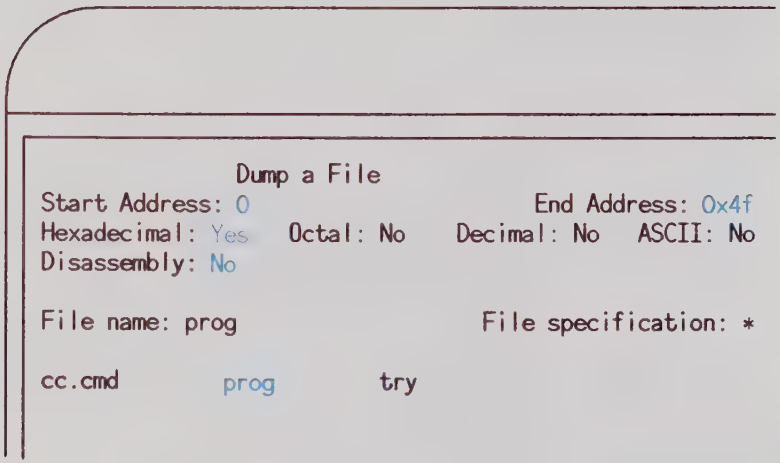
File name Specifies the name of the load file to be dumped. The file can be selected from the list of displayed files. This is done by first selecting the option *File name*, and secondly pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required file. The user can also type the load file name using the character keys.

File specification Specifies that all files will be shown in the menu.

EXAMPLES:

1) The contents of the address range 0-0x4f of the load file *prog* is displayed in hexadecimal format when entering:

```
/fdl
```



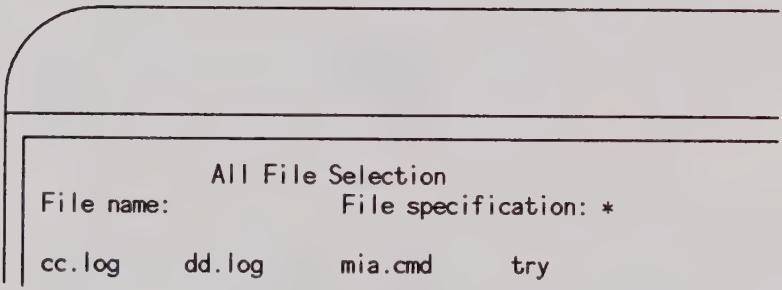
2) In direct command format, this command may be entered as follows:

```
/fdl start=0  end=0x4f  hex=Yes  disa=No  file=prog
```

PATH:

/
Files
Edit
All

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

This command allows the user to edit any type of file without leaving the Debugger. The editor used is *vi* unless the user has specified another editor as described in Chapter 1. The default action of this command is to edit the current default file for the *All* files group. The first time the command is used, no default file name is present. The option for this command allows the user to specify a file name.

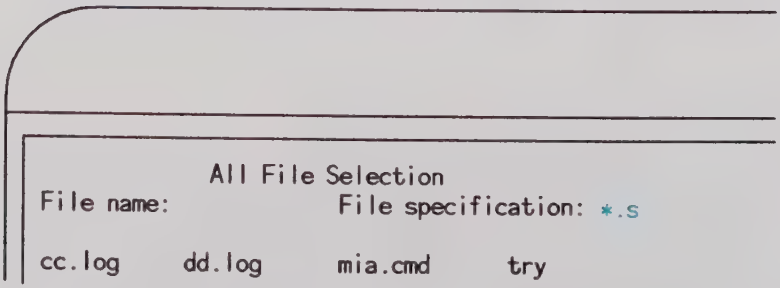
File name Specifies the name of the file to be edited. The file name can be selected from a list of displayed files which are present in the current working directory. This can be done by using the arrow keys to highlight a file name and by subsequently pressing Enter . The user can also type the file name using the character keys.

File specification As no file specification is mentioned by default, all files will be shown in the menu. The user may set the file specification to another wild card pattern.

EXAMPLES:

1) The following command sets the file specification of the *Edit All* command to **.s*.

```
/fea
```



Enter Enter

2) The following command allows the user to select a file which has to be edited:

```
fea
```

```
                All File Selection
File name:                  File specification: *

cc.log      dd.log      mia.cmd      try
```

The file *my.s* in the directory /usr/mia (which is not the current working directory) has to be edited. This file is not present in the list of displayed files, so the user has to type the file name. The space provided beside the keyword *File name* is insufficient. Therefore, the key [Dir Mode] has to be pressed. The user can now enter the full path name.

```
/usr/mia/my.s
```

```
                All File Selection
File name:                  File specification: *

cc.log      dd.log      mia.cmd      try
```

After pressing [Enter] twice, the contents of *my.s* are displayed.

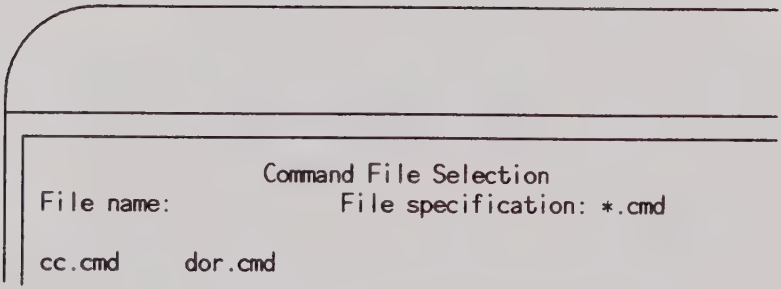
3) In direct command format the previous command is entered as follows:

```
fea file=/usr/mia/my.s
```

PATH:

/
Files
Edit
Command

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

This command allows the user to edit a file matching the default file specification *.cmd* (command file), without leaving the Debugger. The default action of this command is to edit the current default file for the *Files Command* group. The option for this command allows the user to specify a file name.

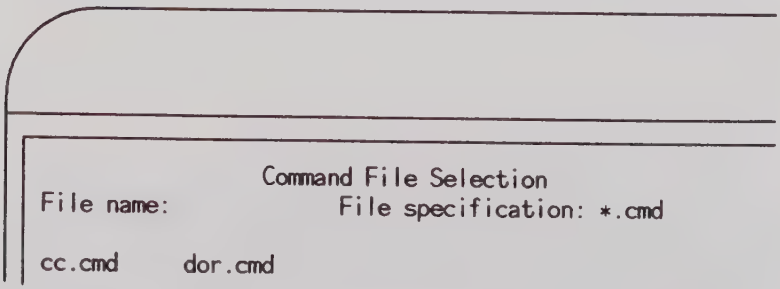
File name Specifies the name of the command file to be edited. The file name can be selected from the list of displayed files, which are the files ending *.cmd* and which are present in the current working directory. This can be done by using the arrow keys to highlight a file name and by subsequently pressing Enter. The user can also type the command file name using the character keys.

File specification Specifies that all file names, suffixed *.cmd*, have to be shown in the menu. The user may alter the file specification.

EXAMPLES:

1) The following command allows the user to select a command file which has to be edited:

```
fec
```



The file *cc.cmd* is displayed when pressing ↓ Enter Enter. The contents of *cc.cmd* can now be edited.

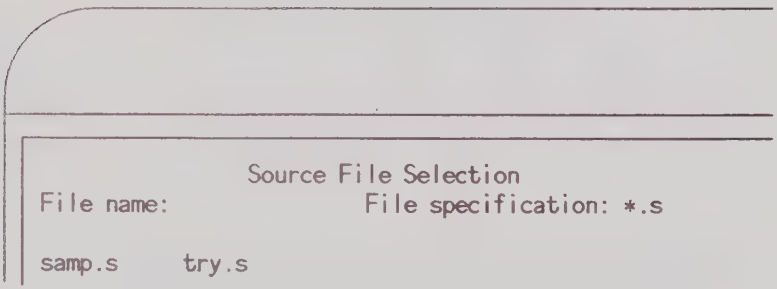
2) In direct command format, the previous command may be issued as follows:

```
/fec file=cc.cmd
```

PATH:

Files
Edit
Source

OPTION WINDOW:



OPTION NAME (direct command format):

file spec

DESCRIPTION:

This command allows the user to edit a file matching the default specification Debugger. The default action of this command is to edit the current default file for the *Files Source* group. The first time the command is used, no default file name is present. The option for this command allows the user to specify a file name.

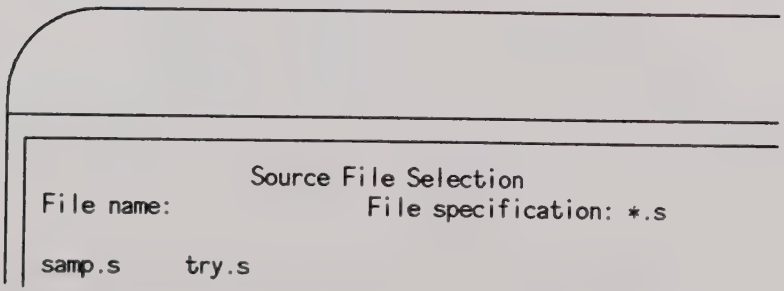
File name Specifies the name of the source file to be edited. The file name can be selected from the list of displayed files, which are the files ending *.s* and which are present in the current working directory. This can be done by using the arrow keys to highlight a file name and by subsequently pressing Enter . The user can also type the source file name using the character keys.

File specification Specifies that all source file names, ending with the suffix *.s*, have to be shown in the menu.

EXAMPLES:

1) The following command allows the user to select a source file which has to be edited:

```
/fes
```



The file *try.s* is displayed when pressing ↓ → Enter Enter . The contents of *try.s* can now be edited.

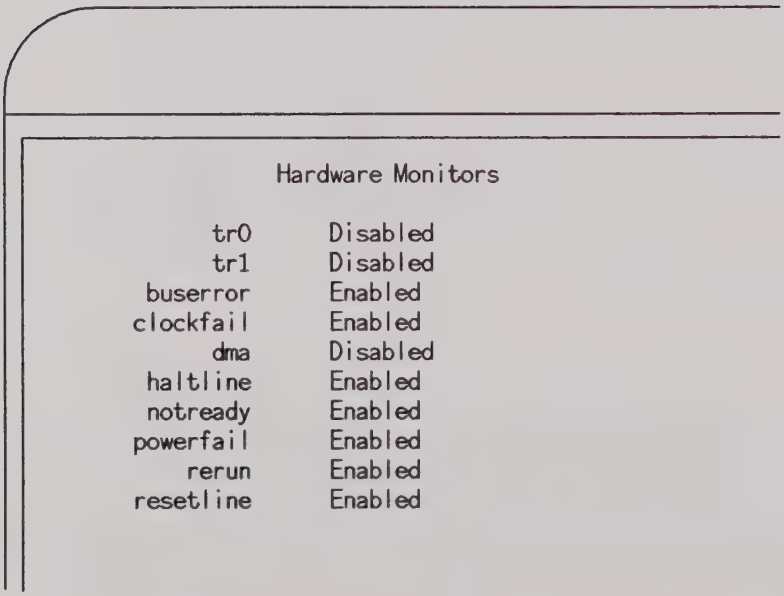
2) In direct command format, the previous command is entered as follows:

```
fes file=try.s
```

PATH:

/
Debugger
HwMonitor
Disable

OPTION WINDOW:



OPTION NAMES (direct command format):

tr0	dma	rerun
tr1	haltline	resetline
buserror	notready	
clockfail	powerfail	

DESCRIPTION:

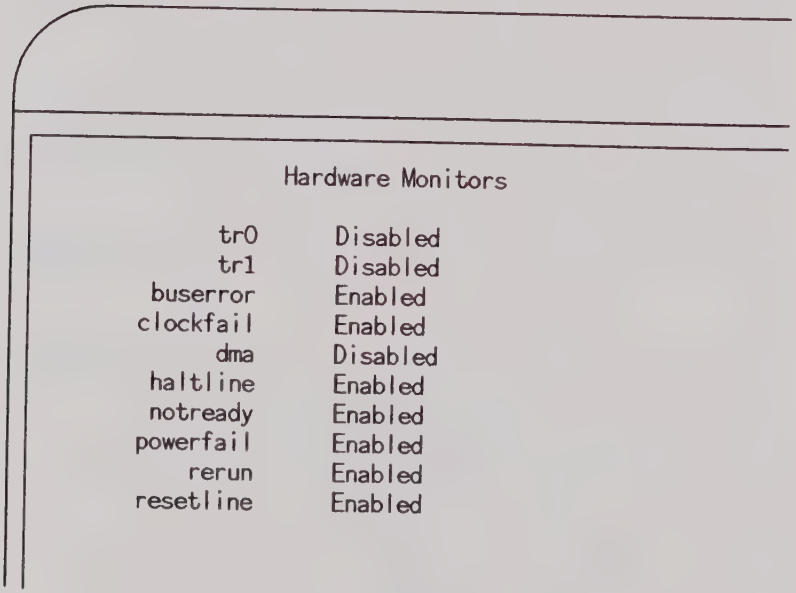
This command allows the user to *enable* or *disable* hardware monitors displayed in the option window. Note that *qr0* and *qr1* are not present in the option list above as they always are *enabled.* However, when their action is set to *pulse* and they are not used for qualified tracing, the action has no effect.

All hardware monitors are explained in the *hwmonitor show* command.

EXAMPLES:

1) The following command shows all hardware monitors and their current status, which is either enabled or disabled.

```
/dhd
```



Press ↓, + or − to set the status of *tr1* to Enabled. Subsequently press Enter twice to execute the command.

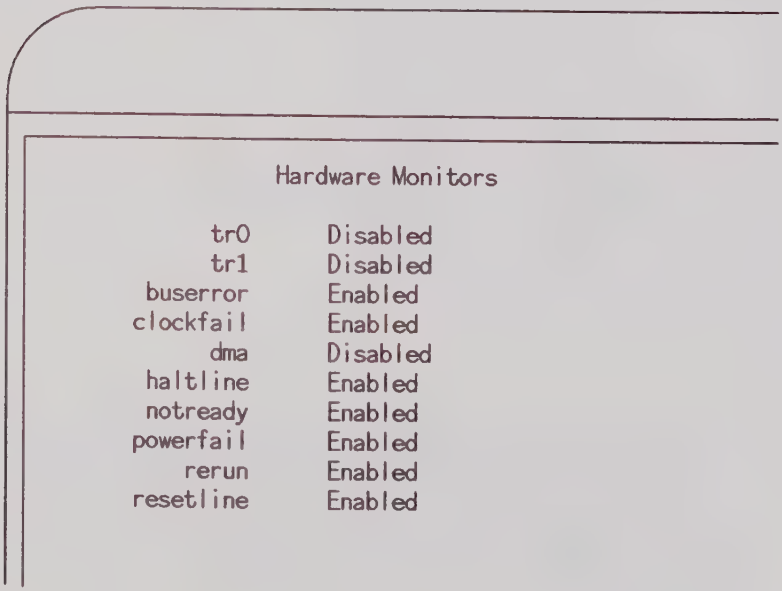
2) In direct command format, the previous command may be issued as follows:

```
/dhd tr1=enabled
```

PATH:

/
Debugger
HwMonitor
Enable

OPTION WINDOW:



OPTION NAMES (direct command format):

tr0	dma	rerun
tr1	haltline	resetline
buserror	notready	
clockfail	powerfail	

DESCRIPTION:

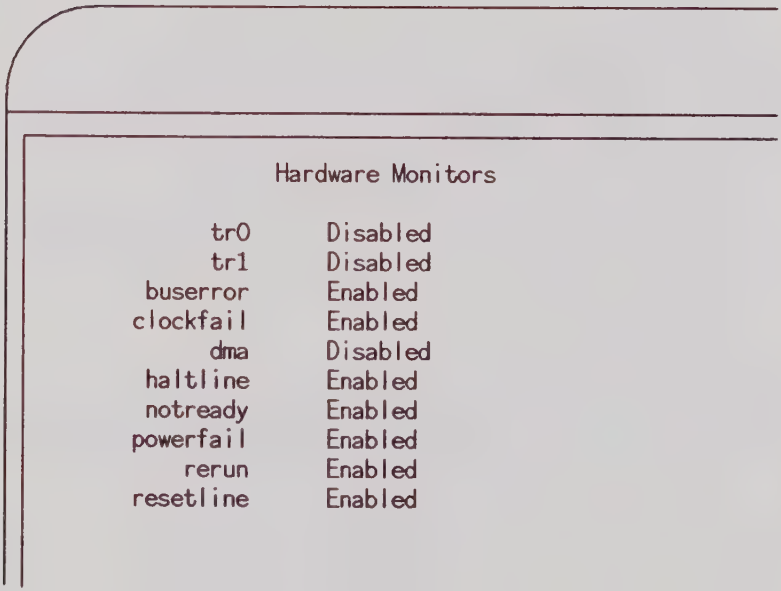
This command allows the user to *enable* or *disable* hardware monitors displayed in the option window. Note that *qr0* and *qr1* are not present in the option list above as they always are *enabled*. However, when their action is set to *pulse* and they are not used for qualified tracing, the action has no effect.

All hardware monitors are explained in the *hwmonitor show* command.

EXAMPLES:

1) The following command shows all hardware monitors and their current status, which is either enabled or disabled.

```
/dhe
```



Press ↓, + or − to set the status of *tr1* to Enabled. Subsequently press Enter twice to execute the command.

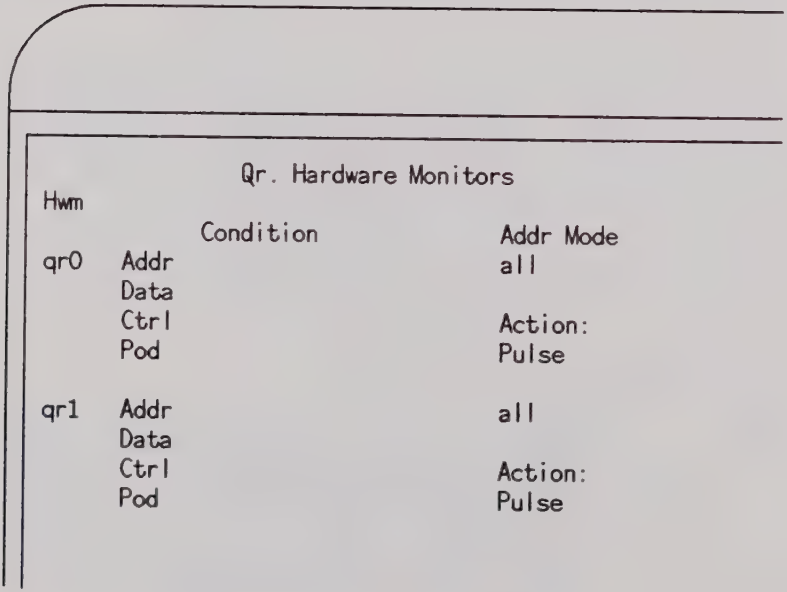
2) In direct command format, the previous command may be issued as follows:

```
/dhe tr1=enabled
```

PATH:

/
Debugger
HwMonitor
Q-Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

qr0.address	qr1.address
qr0.addr_mode	qr1.addr_mode
qr0.data	qr1.data
qr0.pod	qr1.pod

qr0.ctrl	qr1.ctrl
qr0.action	qr1.action

DESCRIPTION:

The *HwMonitor Q-Modify* command can be used to assign a value to the fields of the specified hardware monitors *qr0* and *qr1*. All unaltered fields retain their current value.

The hardware monitors *qr0* and *qr1* consist of a condition part and an action. The condition parts of these monitors include 16 or 24 address lines, 16 data lines, 4 control lines (eg read or write action, program reference, interrupt) and 8 or 0 test-pod lines. The precise number of address and test-pod lines depends on the availability of the Break Trace Expansion cable. If this is installed, then 24 address lines may be specified but no test-pod lines. Otherwise, only 16 address and 8 test-pod lines may be specified.

The fields have the following meanings:

Addr Defines an address to be used when performing system bus match comparisons. When this argument is not altered, the current settings of the address lines remain valid. The x's to which the argument is set by default, denote don't cares. Unless performing single-byte operand comparisons, the address should be even.

Data Specifies the match-value of the target system data bus. When this argument is not altered, the current settings of the data lines remain valid.

Pod

Specifies the match value of the test pod0 lines. If used when the specified pod is absent, an appropriate message will be given. When not altered, the current settings of the test pod0 lines remain valid.

Ctrl

Specifies the match-value of the control lines of the target microprocessor. When not altered, the current settings of the control lines remain valid. Four control lines are definable in this parameter:

Memory or I/O cycle (M/IO)
This line is always *1* in the 68000 debugger.

Write or Read cycle (W/R)

Program Reference (PREF)

Interrupt Acknowledge (INA)

Each control line can be set to 1, 0, x or X, where both x and X denote don't cares.

The meaning of the control-keywords is illiustrated in the following table:

CONTROL LINE VALUES

Keyword	m/io	w/r	pref	ina	Description
pref	x	x	1	x	Program Ref. *up* or *sp*
m	1	x	x	x	Memory Access
mr	1	0	x	x	Memory Read
mw	1	1	x	x	Memory Write
ina	x	x	x	1	Interrupt Acknowledge

Addr Mode Specifies the mode(s) in which the address must be matched, and can be one of the following:

sp	supervisor program
sd	supervisor data
up	user program
ud	user data
all	up, ud, sp, sd
user	up, ud
data	ud, sd
program	up, sp
supervisor	sp, sd

By default, *addr mode* is set to all.

Action Specifies the action to be taken when the specified condition is met. The following actions may be selected by means of the + or − keys:

break If software monitors are triggered by the specified hardware monitor, the evaluation of software monitors is performed. Otherwise, the emulation run is halted.

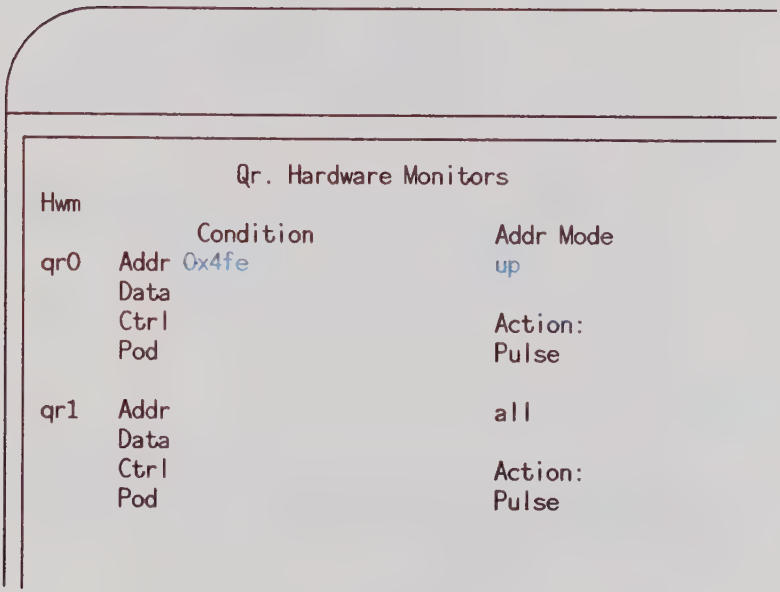
pulse A pulse is generated for the real-time trace mechanism. A break action implies a pulse action.

pulse is the default action.

EXAMPLES:

1) The following command specifies that qr0 will generate a pulse for the real-time trace mechanism when address 0x4fe in *up* has been matched.

```
/dhq
```


Qr. Hardware Monitors
Hwm
Condition
Addr Mode
qr0
Addr 0x4fe
up
Data
Ctrl
Action:
Pod
Pulse
qr1
Addr
all
Data
Ctrl
Action:
Pod
Pulse

Enter Enter

```
hwmon qr0 = up 0x0004fe, "xxxxxxxxxxxxxxxx", "xxxx",
            pulse
hwmon qr0 : enabled
hwmon qr1 = all "xxxxxxxxxxxxxxxxxxxxxxxx", "xxxxxxxxx...
            pulse
hwmon qr1 : enabled
```

2) In direct command format the previous command has to be entered as follows:

```
/dhq qr0.addr_mode up qr0.data =0x4fe qr0.action pulse
```

PATH:

/
Debugger
HwMonitor
Show

DESCRIPTION:

These commands display the settings of all hardware monitors.

The hardware monitors have the following meanings:

tr0 The contents of the target system bus are as specified by the hardware monitor tr0.

tr1 The contents of the target system bus are as specified by the hardware monitor tr1.

. .

qr0 The contents of the target system bus are as specified by the hardware monitor qr0.

qr1 The contents of the target system bus are as specified by the hardware monitor qr1.

buserror A break is generated when the buserror line is active (low).

clockfail A break is generated whenever the clock-low time or the clock-high time >10 micro-secs.

dma A break is generated when the bus request (DMA) line becomes active (low).

haltline A break is generated when the microprocessor halt line is activated (low).

notready A break is generated when the DTACK line is held high for longer than 9ms after the falling edge of the S4 clock pulse.

powerfail A break is generated whenever the target system power supply deviates from the following limits: <4.5 volts or >5.5 volts

rerun A break is generated when the buserror and halt lines are simultaneously active.

resetline A break will be generated whenever the microprocessor RESET line is activated (low).

EXAMPLES:

1) After entering the following command, all hardware monitors and their current settings are shown:

```
/dh
```

```
hwmon tr0 = 1, all 0x5a, ...
            break
hwmon tr0 : disabled, current match counter: 1
hwmon tr1 = 5, user 0x1fe, ...
            pulse
hwmon tr1 : enabled, current match counter: 5
      .
      .
      .
```

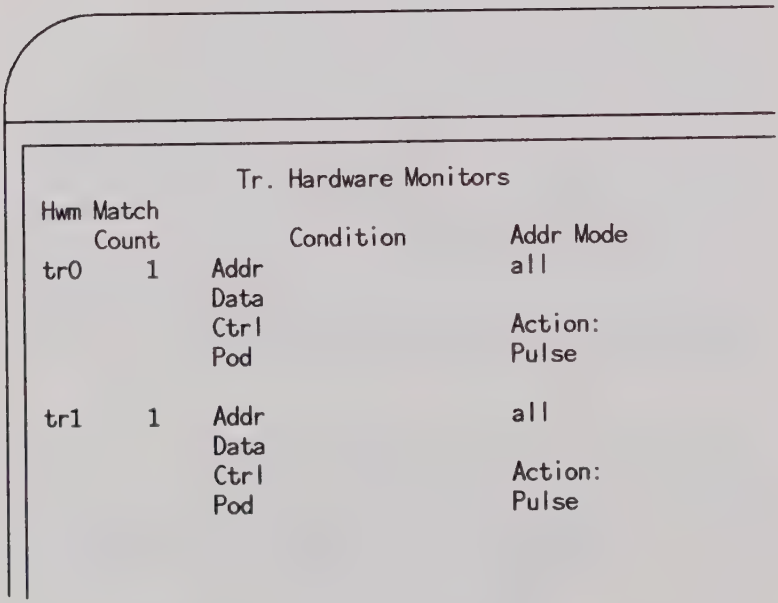
2) In direct command format, the previous command is entered as follows:

```
/dhs
```

PATH:

/
Debugger
HwMonitor
T-Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

tr0.match_count	tr1.match_count
tr0.address	tr1.address
tr0.addr_mode	tr1.addr_mode
tr0.data	tr1.data
tr0.pod	tr1.pod

tr0.ctrl	tr1.ctrl
tr0.action	tr1.action

DESCRIPTION:

The *HwMonitor T-Modify* command can be used to assign a value to the fields of the specified hardware monitors *tr0* and *tr1*. All unaltered fields retain their current value.

The hardware monitors *tr0* and *tr1* consist of a match counter, a condition part and an action. The match counter counts the number of times that the specified condition part is met. The condition parts of these monitors include 16 or 24 address lines, 16 data lines, 4 control lines (eg read or write action, program reference, interrupt) and 8 or 0 test-pod lines. The precise number of address and test-pod lines depends on the availability of the Break Trace Expansion unit. If this is installed, then 24 address lines may be specified but no test-pod lines. Otherwise, only 16 address and 8 test-pod lines may be specified. Note that after setting *tr0* or *tr1*, the relevant hwmonitor is automatically enabled.

The fields have the following meanings:

Match Count Determines how many times a condition must be specified before any action is taken. The value of the *match-counter* must be within the range 1 to 255; initially it is set to 1.

Addr Mode Specifies the mode(s) in which the address must be matched, and can be one of the following:

sp	supervisor program
sd	supervisor data
up	user program

ud	user data
all	up, ud, sp, sd
user	up, ud
data	ud, sd
program	up, sp
supervisor	sp, sd

The default *addr mode* is all.

Addr Defines an address to be used when performing system bus match comparisons. When this argument is not altered, the current settings of the address lines remain valid. The x's to which the argument is set by default, denote don't cares. Unless performing single-byte operand comparisons, the address should be even.

Data Specifies the match-value of the target system data bus. When this argument is not altered, the current settings of the data lines remain valid.

Ctrl Specifies the match-value of the control lines of the target microprocessor. When not altered, the current settings of the control lines remain valid. Four control lines are definable in this parameter:

Memory or I/O cycle (M/IO)
This line always *1* in the 68000 debugger.

Write or Read cycle (W/R)

Program Reference (PREF)

Interrupt Acknowledge (INA)

Each control line can be set to 1, 0, x or X, where both x and X denote don't cares.

The meaning of the control-keywords can be obtained from the following table:

CONTROL LINE VALUES

Keyword	m/io	w/r	pref	ina	Description
pref	x	x	1	x	Program Ref. *up* or *sp*
m	1	x	x	x	Memory Access
mr	1	0	x	x	Memory Read
mw	1	1	x	x	Memory Write
ina	x	x	x	1	Interrupt Acknowledge

Pod

Specifies the match value of the test pod0 lines. If used when the specified pod is absent, an appropriate message will be given. When not altered, the current settings of the test pod0 lines remain valid.

Action

Specifies the action to be taken when the specified condition is met. The following actions may be selected by means of the + or − keys:

break If software monitors are triggered by the specified hardware monitor, the evaluation of software monitors is performed. Otherwise, the emulation run is halted.

break is the default action.

pulse If the specified hardware monitor is *tr0* or *tr1*, a pulse is generated which can be used by external equipment connected to the two BNC connectors on the MAB. A *break* action implies a *pulse* action.

EXAMPLES:

1) This command specifies that tr0 will generate a break if the data-value 0x4e71 has been matched 5 times when working in *user* mode.

```
/dht
```

```
                    Tr. Hardware Monitors
Hwm Match
    Count              Condition          Addr Mode
tr0     5  Addr                          user
           Data  0x4e71
           Ctrl                          Action:
           Pod                           Pulse

tr1     1  Addr                          all
           Data
           Ctrl                          Action:
           Pod                           Pulse
```

Enter Enter

```
hwmon tr0 = 5, user "xxxxxxxxxxxxxxxx", 0x0004e71, ...
            pulse
hwmon tr0 : enabled, Current match counter : 5
```

2) In direct command format, the previous command has to be entered as follows:

```
dht tr0.match_count=5 tr0.addr_mode=user tr0.data=0x4e71
```

On the command line all option names are optional. Therefore, we have specified only those options that are given a value other than the default value.

PATH:

/
Config
Interrupt
Disable

DESCRIPTION:

This command disables the passing of all interrupts from the target microproocessor to the emulator.

EXAMPLES:

1) All interrupts will be disabled after entering the following command:

/cid
Enter

```
interrupts : disabled
```

PATH:

/
Config
Interrupt
Enable

DESCRIPTION:

This command enables all interrupts to be passed from the target microproocessor to the emulator. Interrupts are enabled at invokation of the debugger.

EXAMPLES:

1) All interrupts will be enabled after entering the following command:

```
/cie
Enter

interrupts : enabled
```

PATH:

/
Config
Interrupt
Show

DESCRIPTION:

This command shows whether the passing of the interrupts from the target microprocessor to the emulator is disabled or enabled.

EXAMPLES:

1) The following command shows that all interrupts are currently enabled:

/ci
Enter

```
interrupts : enabled
```

PATH:

/
Services
Level
Cycle

FUNCTION KEYS:

Cycle Lvl

DESCRIPTION:

The default window is cycled around the levels of the current default window, displaying each level as it is selected.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and two display windows. Window 2 has four levels and is the default window. Level 4 is displayed.

   Window 2 level 1 becomes the default window, when entering:

   /slc
   Enter

PATH:

/
Services
Level
Delete

FUNCTION KEYS:

Del Lvl

DESCRIPTION:

This command deletes the displayed level of the default window. If this level is the last remaining level of a window, the window will be deleted. The last remaining level of the last remaining display window cannot be deleted.

EXAMPLES:

1) Assume that the basic screen consists of the menu window and two display windows, each having 3 levels: Window 1 Level 2 is the default window.

   Level 2 and 3 of Window 1 can be deleted by entering the command /sld twice.

PATH:

/
Services
Level
Make

FUNCTION KEYS:

Make Lvl

DESCRIPTION:

This command creates a new level of the default window. The new level takes over the display area of the default window, and is initially empty. The original level, being displayed in the default window before creating the new level, is preserved. This means that the original level still exists and the data seen through the level is unaffected but no longer displayed.

Note that for each window, a maximum of four levels can be created.

EXAMPLE:

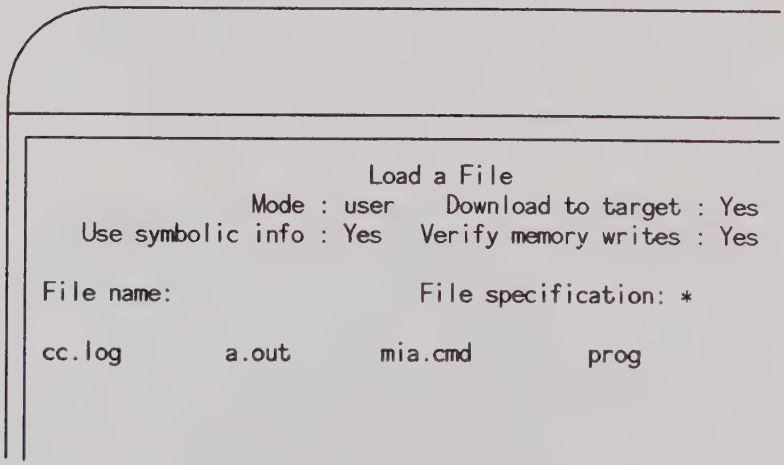
1) Assume that the basic screen currently consists of a menu window and 1 display window which has one level.

   A new level can be created by issuing the command /slm. Level 1 will be preserved.

PATH:

/
Files
Load
All

OPTION WINDOW:



OPTION NAMES (direct command format):

mode	target
symbols	verify
file	spec

DESCRIPTION:

The command loads an out.h format load file into memory. The default action for this command is to load the current default file

for the *All* files group. The following options can be specified by the user:

Mode

The + or − keys can be pressed to set the processor mode to be used for down loading, to either *user* or *supervisor*. By default *Mode* is set to *user*.

Download to target

When this option is set to *Yes*, which is the default value, object code will be loaded into memory. When setting the option to *No* by pressing either the + or − key, no object code will be loaded into memory.

Use symbolic info

This option can be set to *Yes* or *No* by means of the + or − keys. When it is set to the default value *Yes*, the symbol table is loaded into the Debugger table. Otherwise, the loading of the symbol table is suppressed.

Verify memory writes

By default, this option is set to *Yes* which means that loaded data is verified each time object code is written to memory. Pressing the + or − key sets the option to *No*. In this case, the loaded code is not to be verified after a transfer of data from disc to memory.

File name

Specifies the name of the file to be loaded. The file name can be selected from the list of displayed files, which are the files present in the current working directory. This is done by first selecting the option *File name*, and secondly, pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required file. The user can also type the load file name

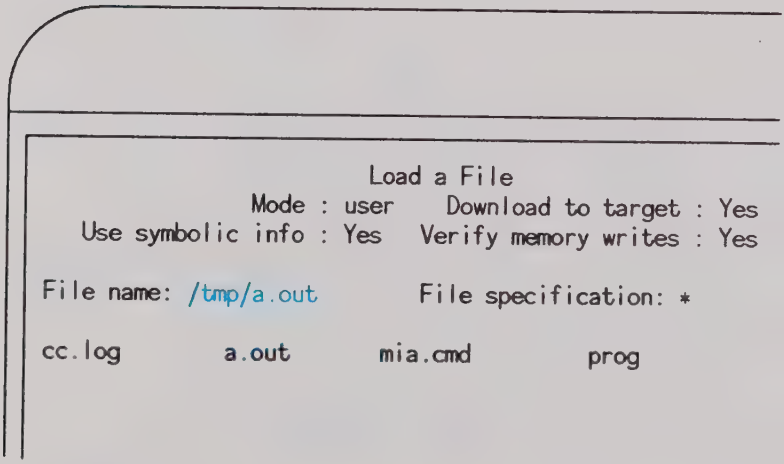
using the character keys.

File specification Specifies that any load file may be selected. The user may alter the file specification.

EXAMPLES:

1) The file */tmp/a.out* is loaded:

```
fla
```



Enter

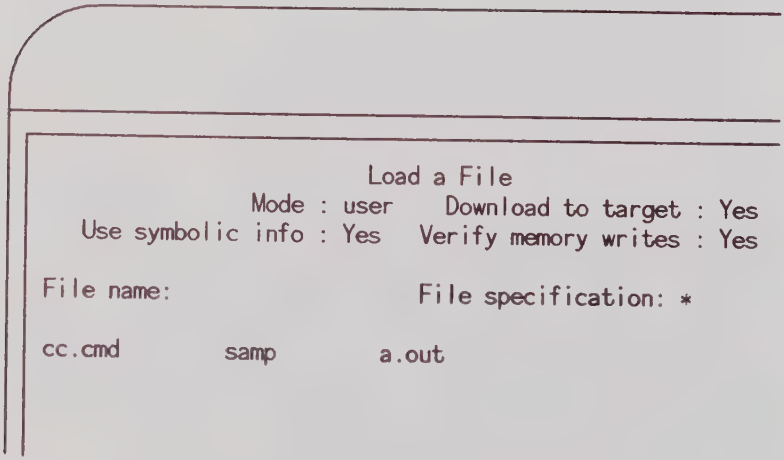
2) In direct command format the previous command may be entered as follows:

```
/fla file=/tmp/a.out
```

PATH:

/
Files
Load
Load

OPTION WINDOW:



OPTION NAMES (direct command format):

mode	target
symbols	verify
file	spec

DESCRIPTION:

This command loads an out.h format load file into memory. The default action for this command is to load the current default file

for the *load files* group. The following options can be specified by the user:

Mode The + or − keys can be pressed to set the processor mode to be used for down loading, to either *user* or *supervisor*. By default *Mode* is set to *user*.

Download to target When this option is set to *Yes*, which is the default value, object code will be loaded into memory. When setting the option to *No* by pressing a + or − key, no object code will be loaded into memory.

Use symbolic info This option can be set to *Yes* or *No* by means of the + and − keys. When it is set to its default value *Yes*, the symbol table is loaded into the Debugger table. Otherwise, the loading of the symbol table is suppressed.

Verify memory writes By default, this option is set to *Yes* which means that loaded data is verified each time object code is written to memory. Pressing either the + or − key sets the option to *No*. In this case, the loaded code is not to be verified after a transfer of data from disc to memory.

File name Specifies the name of the file to be loaded. The file name can be selected from the list of displayed files, which are the files present in the current working directory. This is done by first selecting the option *File name*, and secondly, pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required

file. The user can also type the load file name using the character keys.

File specification Specifies that any load file may be selected. The user may alter the file specification.

EXAMPLES:

1) The file */tmp/prog* is loaded:

```
fll
```

```
                    Load a File
            Mode : user     Download to target : Yes
  Use symbolic info : Yes   Verify memory writes : Yes

File name: /tmp/prog          File specification: *

cc.cmd        samp        a.out
```

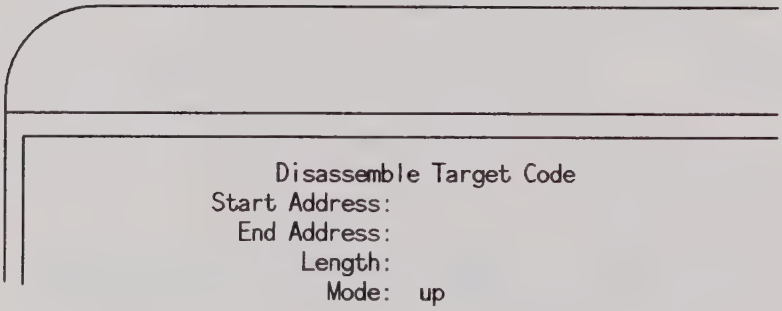
2) In direct command format the previous command is entered as follows:

```
/fll file=/tmp/prog
```

PATH:

/
Target
Memory
Disasm

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
length	addr_mode

DESCRIPTION:

This command displays a program disassembly of a specified memory range.

Start Address Specifies the first location of the memory range.

End Address Specifies the last location of the memory range.

Length Specifies the number of locations of which the memory range consists.

It is not possible to specify the parameters end address and length at the same time.

Only two of the parameters explained above have to be specified.

Mode *up*, *sp*, *ud* or *sd* may be selected from a select-table range of modes.

EXAMPLES:

1) The following command specified that a program disassembly of the memory range 0x30 to 0x40 in *up* mode has to be displayed:

```
tmd
```

```
Disassemble Target Code
Start Address: 0x30
  End Address: 0x40
       Length:
         Mode: up
```

After pressing Enter , the required program disassembly is displayed.

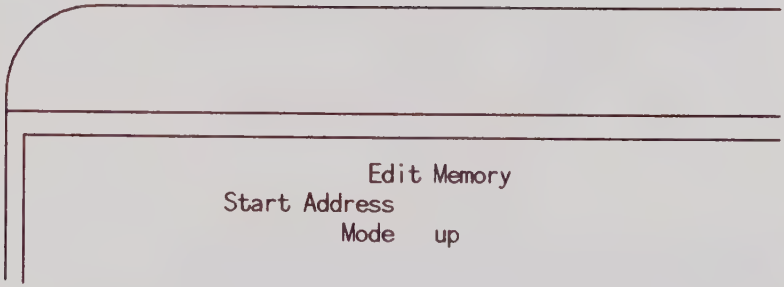
2) In direct command format the previous command may be entered as follows:

```
/tmd start=0x30 end=0x40 addr_mode=up
```

PATH:

/
Target
Memory
Edit

OPTION WINDOW:



OPTION NAMES (direct command format):

start addr_mode

DESCRIPTION:

This command allows the user to display an area of memory in the default display window. After making the display window the new current window, this memory area may be edited.

The following options can be specified.

Start Address Specifies the first address of the memory area to be displayed.

Note

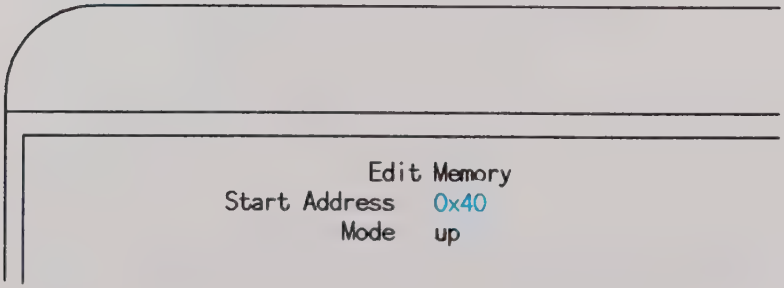
As the whole display area of the default display window is used to show memory locations, the end address is determined by the space of the display area and not by the user.

Mode *up*, *ud*, *sp* or *sd* can be selected from a selecttable range of modes.

EXAMPLES:

1) After entering the following command, the default display window is filled with the memory locations starting at address 0x40.

/tme



Enter

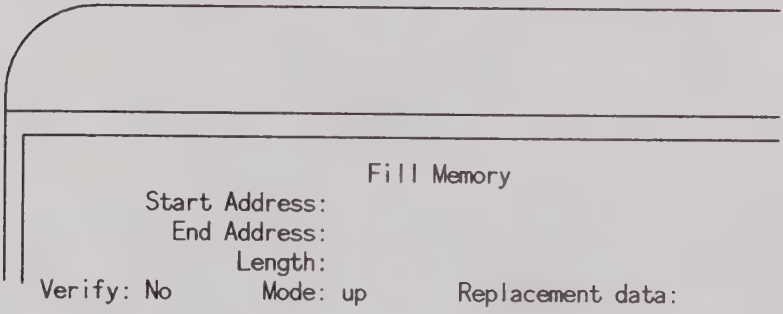
Press the key Next Wnd to make the display window the

current window. The contents of the memory locations may now be edited.

PATH:

/
Target
Memory
Fill

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
length	verify
addr_mode	repl

DESCRIPTION:

This command sets the contents of the specified memory locations to the specified values. One value, a series of values or a pattern of values can be entered to set the specified memory range.

The following options are provided:

Start Address Specifies the first memory location to be altered.

End Address Specifies the last memory location to be altered.

Length Specifies the number of locations (length) of the memory range to be altered.

If the user specifies only a start address and one value, the memory location specified by start address will be set to the value.

If the user specifies only a start address and a series of values, the memory location specified by start address and those following will be set to the values specified.

If the user specifies an address range and a pattern of values, the pattern will be repeated within the specified range.

Verify If this option is set to its default value *No*, the contents of memory locations are not verified when writing. Pressing a + or − key sets the option to *Yes*, in which case the contents of memory locations are verified when writing.

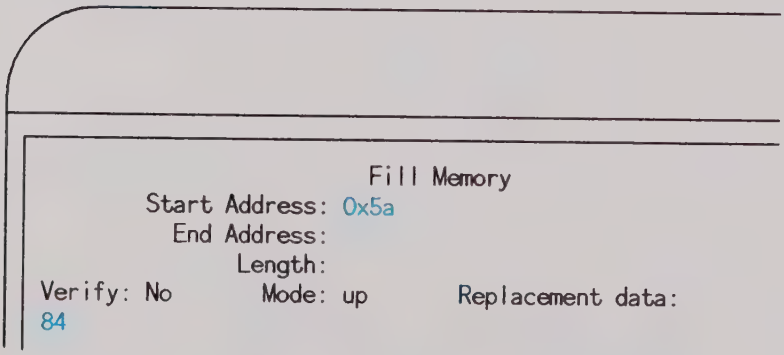
Mode One of the modes *up*, *ud*, *sp* and *sd* can be selected by means of a + or − key. The default is *up*.

Replacement Data One or more value(s) may be entered.

EXAMPLES:

1) The contents of the memory location 0x5a are set to the decimal value 84:

/tmf



Enter

2) The contents of the memory locations 0x10 to 0x13 are set to 0, 0xf, 0, 0xf.

```
                    Fill Memory
        Start Address: 0x10
          End Address: 0x13
               Length:
Verify: No       Mode: up       Replacement data:
0 0xf
```

Enter

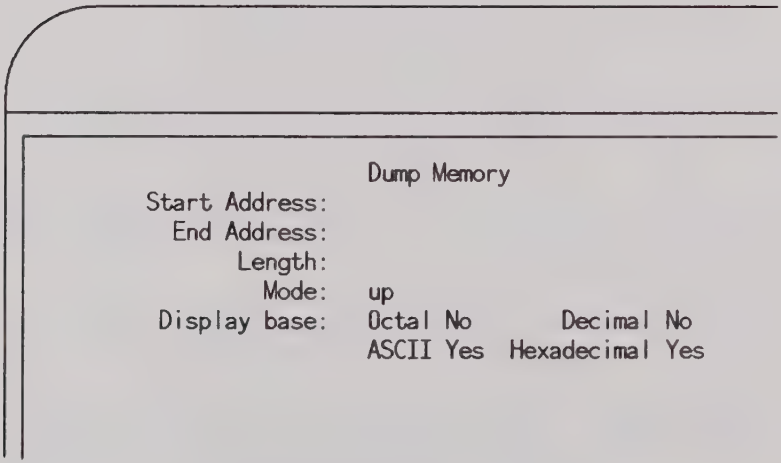
3) In direct command format the previous command may be entered as follows:

```
mmf start 0x10 end: 0x13 verify No addr_mode up repl 0 0xf
```

PATH:

/
Target
Memory
MemDump

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
length	addr_mode
octal	decimal
ascii	hex

DESCRIPTION:

This command displays an area of memory in the required format. The following options are provided:

Start Address Specifies the first location of the memory range to be displayed.

End Address Specifies the last location of the memory range to be displayed.

Length Specifies the number of locations (length) of the memory range to be displayed.

The entered values are used to form the default values for all other *Memory* commands. It is not possible to specify end address and length at the same time.

Mode The + or − keys may be pressed to select one of the modes *up*, *ud*, *sp* and *sd*.

Display base Octal, ascii, decimal and hexadecimal formats may be selected. The specified memory range is displayed in each format that is set to *Yes*. By default, only the ASCII and hexadecimal format are set to *Yes*. The + or − keys can be used to alter the settings.

EXAMPLES:

1) The contents of the memory locations from 0 to 0x1f are displayed in hexadecimal format.

```
/tmm
```

```
                    Dump Memory
Start Address:      0
  End Address:
       Length:      0x1f
         Mode:      up
Display base:       Octal No        Decimal No
                    ASCII No    Hexadecimal Yes
```

Enter

```
address      |                  hex
-------------------------------------------------------
             |
up 0x000000 cd 20 8e 50 00 9a f0 ...
up 0x000010 ab 27 bd 01 00 3e s7 ...
```

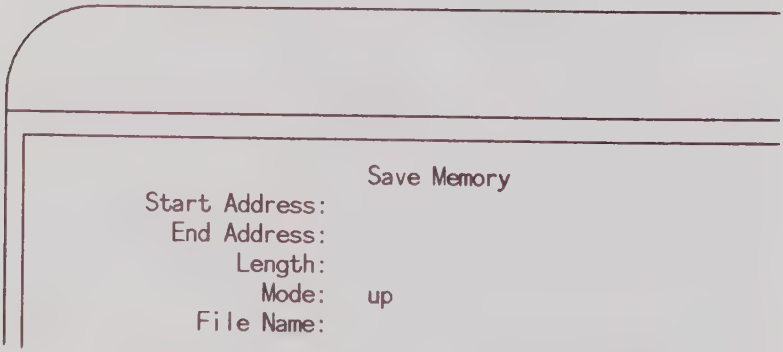
2) In direct command format this command may be entered as follows:

```
/tmm start=0 length=0x1f ascii=No
```

PATH:

/
Target
Memory
Save

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end
length	addr_mode
file	

DESCRIPTION:

This command stores the contents of the specified memory locations in a new section of the specified target load file.

The following options are provided:

Start Address Specifies the first memory location to be altered.

End Address Specifies the last address of the address range to be stored.

Length Specifies the number of memory locations to be stored.

Once entered, these values of Start Address and End Address or length form the default values for the *Memory* command. At any one time, either end address or length can be specified.

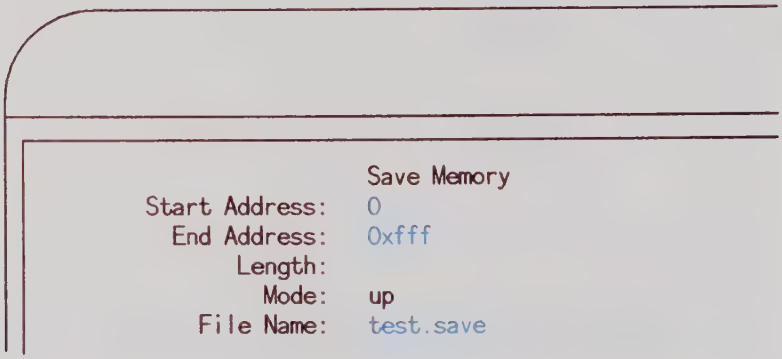
Mode The + or − keys may be pressed to select one of the modes *up*, *ud*, *sp* and *sd*.

File name Specifies the pathname for the target load file. The file will be in *out.h* format and will not contain any debug information or symbol tables.

EXAMPLES:

1) The contents of memory locations 0 to 0xfff are stored in the files test.save

```
/tms
```



Enter

2) In direct command format this command may be entered as follows:

```
/tms  start=0 end=0xfff file=test.save
```

PATH:

/
Config
MemoryMap
Disable

DESCRIPTION:

This command is used to disable the processor mapping mechanism. When the mapping mechanism is disabled, all memory is mapped to target.

EXAMPLES:

1) The mapping mechanism is disabled:

```
/cmd
Enter

map : disabled
```

PATH:

/
Config
MemoryMap
Enable

DESCRIPTION:

This command is used to enable the processor mapping mechanism. When the mapping mechanism is enabled, the selected segment will be mapped as specified at the time of disabling. By default, the mapping mechanism is enabled.

EXAMPLES:

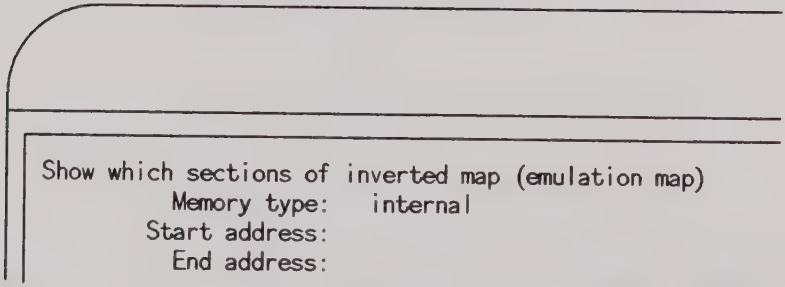
1) The mapping mechanism is enabled:

```
cme
Enter

map : enabled
```

PATH:

/
Config
MemoryMap
Invert

OPTION WINDOW:



OPTION NAMES (direct command format):

memtype start
end

DESCRIPTION:

The command allows the user to display the target processor memory map from the viewpoint of the emulation memory. The default action of this command is to display the map setting(s) of the entire address range of internal and external emulation memory. The available/unavailable emulation memory blocks are also given. Furthermore, it is shown whether or not a processor address range is mapped to an emulation memory block.

The following options allow the user to display the emulation memory map over a chosen range:

Memory type Specifies the memory type of which the map settings have to be shown. It can be one of the following:

internal	indicates internal emulation memory
external	indicates external emulation memory
all	indicates both internal and external memory

Start address Specifies the start address of the emulation memory to be displayed.

End address Specifies the end address of the emulation memory to be displayed.

EXAMPLES:

1) The internal emulation memory map is displayed:

```
/cm
←
Enter Enter
```

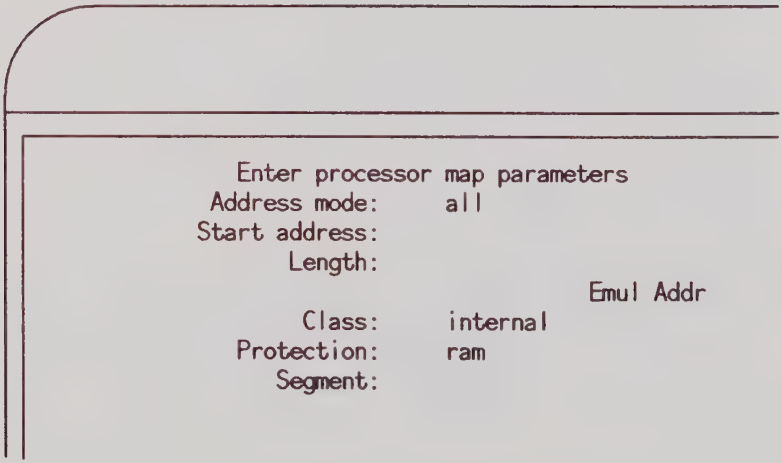
2) In direct command format, the previous command may be entered as follows:

```
/cmi memtype=all start=0 end=0x1fff
```

PATH:

/
Config
MemoryMap
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

addr_mode	start
length	class
prot	emu_addr
segment	

DESCRIPTION:

This command maps the specified address range and address

mode of the processor's memory to the specified *Class* with the specified *protection.* It may also be used to select the current segment.

Only an address range within the current segment may be mapped. When the address range is mapped to internal/external emulation memory and no emulation address is specified, unmapped blocks in emulation memory will be used if possible. Updates are done in units of one block (4 kbytes). The memory modified will be that containing the address range specified.

Address mode May be one of *all*, *user*, *up*, *ud*, *data*, *program*, *sp*, *sd* or *supervisor. All* is the default mode.

Start Address Only the map setting(s) from the specified address onwards, have to be modified.

Length Specifies the length of the address range to be modified.

Class Specifies the map class to be used.

internal	specifies that memory must be mapped to internal emulation memory.
external	specifies that memory must be mapped to external memory.
target	specifies that memory must be mapped to static target memory.

Protection The following protections can be used:

ram specifies that all memory accesses will be allowed.

rom specifies that access to this part of memory will be restricted to read only.

guarded specifies that no memory access is allowed in this part of memory.

Segment The segment, in which the specified address is located, is selected.

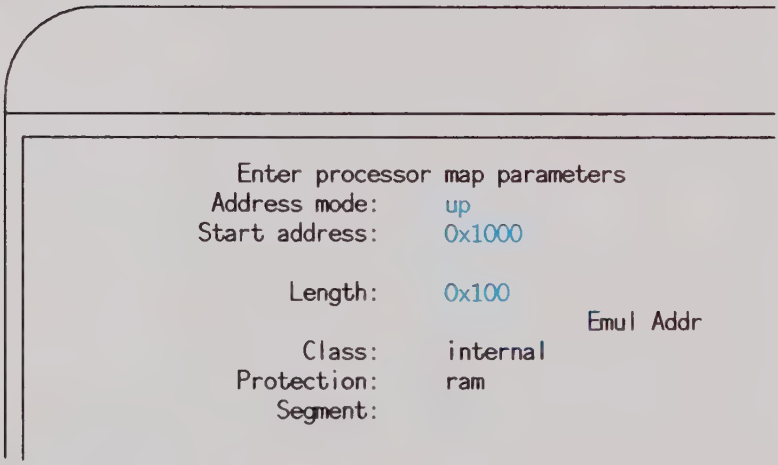
Each segment has a size of 128K, and therefore always starts on a 128K boundary. At any one time, only one segment for *all* modes can be mapped. At the start of a session the default segment is 0 to 0x01ffff.

Emul Addr Specifies the start address of the processor memory range to which emulation memory has to be mapped.

EXAMPLES:

1) The Start Address and length are set to 0x1000 and 0x100. This specifies that only the map setting(s) of the address range 0x1000 - 0x10ff in *up* have to be modified. The actual range modified will be that containing this range, namely 0x1000 - 0x1fff.

```
/cmm
```



2) In direct command format the previous command may be entered as follows:

```
/cmm start=0x1000 length=0x100
```

3) The segment 0x40000 to 0x5ffff can be selected as follows:

```
/cmm
```

```
        Enter processor map parameters
  Address mode:     all
 Start address:
        Length:
                                Emul Addr
         Class:     internal
    Protection:     ram
       Segment:     0x40000
```

Enter Enter

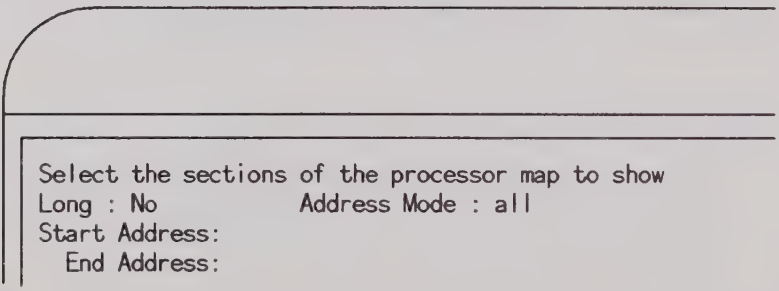
4) In direct command format, the previous command may be entered as follows:

```
/cmm segment=0x40000 class=internal emu_addr=0
```

PATH:

/
Config
MemoryMap
Show

OPTION WINDOW:



OPTION NAMES (direct command format):

long	addr_mode
start	end

DESCRIPTION:

The mapping of the processor memory is displayed.

The options allow selection of the address range and mode(s) of the processor memory map to be shown. They have the following meaning:

Long Specifies whether or not the addresses of the emulation blocks have to be displayed. By

default, no emulation blocks are displayed.

Addr. mode Is one of *up*, *ud*, *sp*, *sd*, *user*, *program*, *supervisor*, *data*, *all*. The default mode is *All*.

Start Address Specifies the start address of the map setting(s) to be displayed.

End Address Specifies the end address of the map setting(s) to be displayed.

EXAMPLES:

1) The following command shows the current mapping of address range 0 - 0xfff of the processor memory:

```
cms

Select the sections of the processor map to show
Long : No              Address Mode : all
Start Address: 0
  End Address: 0xfff
```

Enter Enter

```
map : enabled
map up 0x000000 0x000fff = internal ram
map ud 0x000000 0x000fff = internal ram
map sp 0x000000 0x000fff = internal ram
map sd 0x000000 0x000fff = internal ram
```

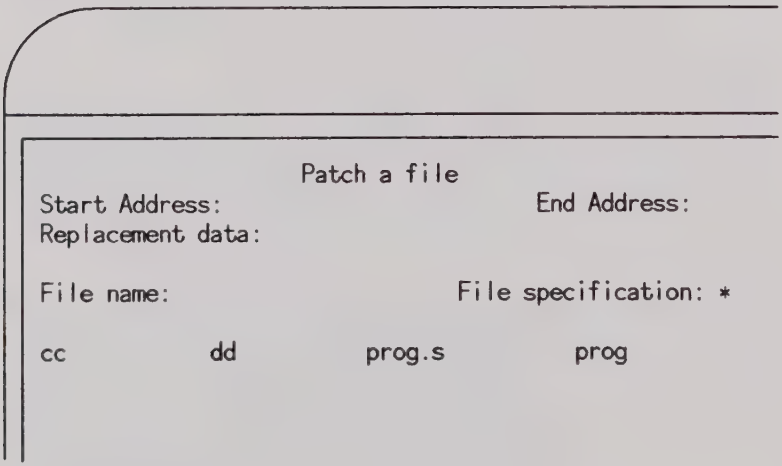
2) In direct command format, the previous command may be entered as follows:

```
/cms start=0 end=0xfff
```

PATH:

/
Files
Patch
All

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end	spec
repl	file	

DESCRIPTION:

This command allows the user to modify a load file. There is no default action for this command. The file name, as well as the address(es), and the corresponding replacement data have to be

specified.

Start Address Specifies the first location address that has to be modified.

End Address Specifies that an address range, beginning at *Start Address* and ending at *End Address* has to be modified. If this option is not specified, only the *Start Address* will be modified.

Replacement data Specifies the values which have to be assigned to the address range specified by *Start Address* and *End Address*. When less data values are specified than addresses, the data sequence is repeatedly stored in the load file until the end of location is reached.

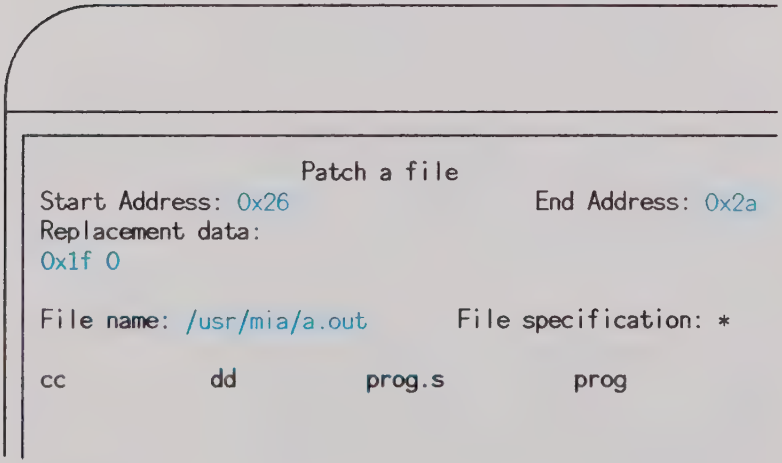
File name Specifies the name of the load file to be modified. The file can be selected from the list of displayed files. This is done by first selecting the option *File name*, and secondly pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required file. The user can also type the load file name using the character keys.

File specification As no file specification is mentioned, any file type may be selected. The file specification may be altered.

EXAMPLES:

1) The following command modifies the contents of the addresses 0x26 to 0x2a. The addresses 0x26, 0x28 and 0x2a are set to 0x1f. The addresses 0x27 and 0x29 are set to 0.

```
fpa
```



Enter

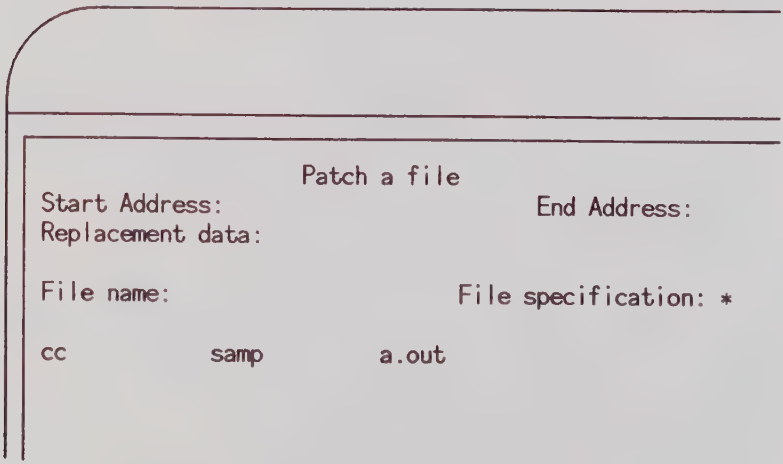
2) In direct command format the previous command may be executed as follows:

```
/fpa start=0x26 end=0x2a file=/usr/mia/a.out repl=0x1f 0
```

PATH:

/
Files
Patch
Load

OPTION WINDOW:



OPTION NAMES (direct command format):

start	end	spec
repl	file	

DESCRIPTION:

This command provides the capability of modifying a load file. There is no default action for this command. The file name, as well as the address(es) and the corresponding replacement data

have to be specified.

Start Address Specifies the first location address that has to be modified.

End Address Specifies that an address range, beginning at *Start Address* and ending at *End Address* has to be modified. If this option is not specified, only the *Start Address* will be modified.

Replacement data Specifies the values which have to be assigned to the address range specified by *Start Address* and *End Address*. When less data values are specified than addresses, the data sequence is repeatedly stored in the load file until the end of location is reached.

File name Specifies the name of the load file to be modified. The file can be selected from the list of displayed files. This is done by *first* selecting the option *File name*, and secondly pressing the arrow down key to highlight the first file in the list. The arrow keys can now be used to select the required file. The user can also type the load file name using the character keys.

File specification As no file specification is mentioned, any file type may be selected. The user may set the file specification to any wild card pattern.

EXAMPLES:

1) The following command modifies the contents of the addresses 0x20 and 0x21 to 0x9 and 0x1a respectively:

```
fpl
```

```
                    Patch a file
Start Address: 0x20                    End Address: 0x21
Replacement data:
0x9 0x1a

File name: samp                    File specification: *

cc          samp          a.out
```

Enter

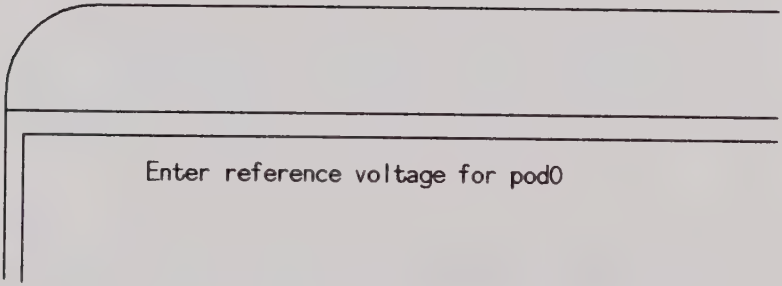
2) In direct command format the previous command may be entered as follows:

```
/fpl start=0x20 end=0x21 file=samp repl=0x9 0x1a
```

PATH:

/
Config
Pod
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

volt

DESCRIPTION:

This command allows the user to specify a threshold voltage value for pod0. This value must be a number in the range of 0.1 to 10.9 volts in steps of 0.1

EXAMPLES:

1) Set the threshold voltage to 3.2:

```
cpm

Enter reference voltage for pod0 3.2
```

After pressing Enter twice, the new value of pod0 is displayed.

2) In direct command format, the previous command is entered as follows:

```
/cpm volt=3.2
```

PATH:

/
Config
Pod
Show

DESCRIPTION:

This command displays the operating threshold voltage for pod0. The threshold voltage is the voltage above which a logic '1' is recognised; anything less than this voltage is recognised as a logic '0'.

EXAMPLES:

1) This command displays the value of the threshold voltage for pod0:

```
/cp
Enter
```

```
pod0 = "1.3"
```

2) In direct command format, the previous command is entered as follows:

```
cps
```

PATH:

/
Target
Processor
Halt

DESCRIPTION:

When issuing this command, the processor stops executing the user's program. The value of the current trace window setting determines the precise moment at which execution will be halted.

EXAMPLES:

1) When a program runs in an endless loop, program execution may be halted by entering the following command:

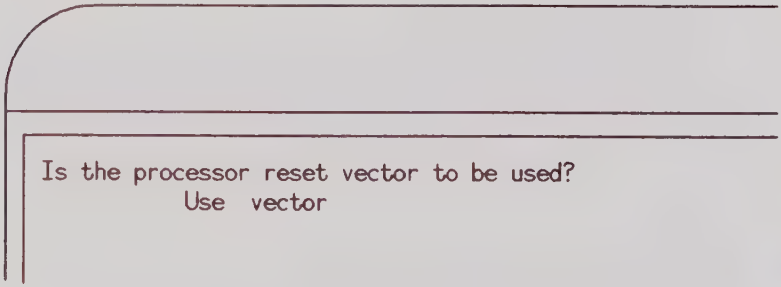
```
/tph
Enter
```

```
Emulation halted
```

PATH:

/
Target
Processor
P-Reset

OPTION WINDOW:



OPTION NAMES (direct command format):

vector

DESCRIPTION:

The *P-Reset* command resets the target processor. The default action of this command is to reset the target processor using the hardware reset vector. The reset vector is located at addresses 0 to 7 in *sp* mode. One option can be used to select *novector*, which means that the reset vector is no longer used to reset the target processor.

EXAMPLES:

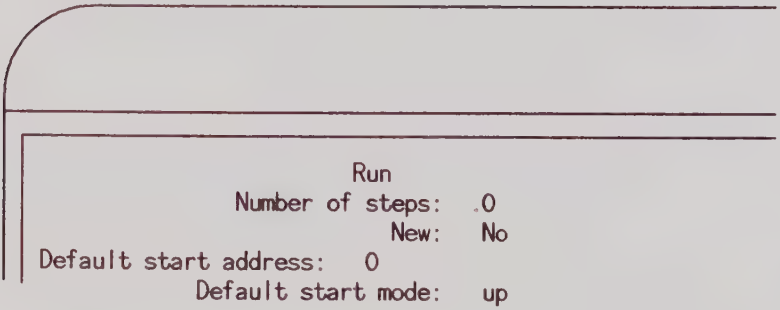
1) The target processor is reset:

```
/tp
→ → Enter
```

PATH:

/
Target
Processor
Run

OPTION WINDOW:



OPTION NAMES (direct command format):

start	step
new	mode

DESCRIPTION:

This command starts an emulation run. By default, program execution starts at the address determined by the current value of the program counter. The following options can be specified:

Number of steps Specifies the number of steps after which program execution must be halted. One step is completed each time an execution point is reached where at least one action list of a software monitor is executed.

New Specifies that program execution starts at the default start location. Unless specified otherwise this is 0. Before starting execution, the real time trace memory is cleared and the real time event counter is set to 0.

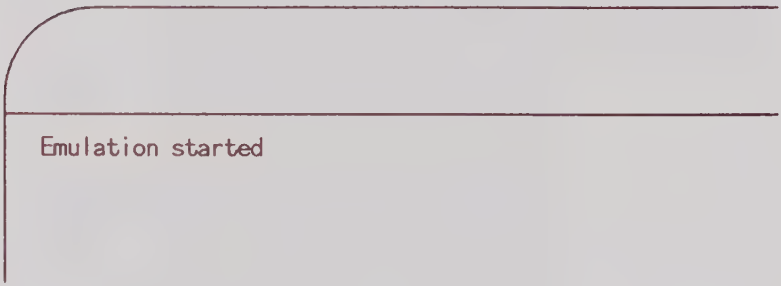
Default start address Specifies the address at which program execution has to be started when New is set to *Yes*.

Default start mode Specifies the mode, which is either *up* or *sp*, in which the program has to be run. The default mode is *up*.

EXAMPLES:

1) The following command causes program execution to start at the address specified by the current content of the program counter.

```
/tpr
Enter
```



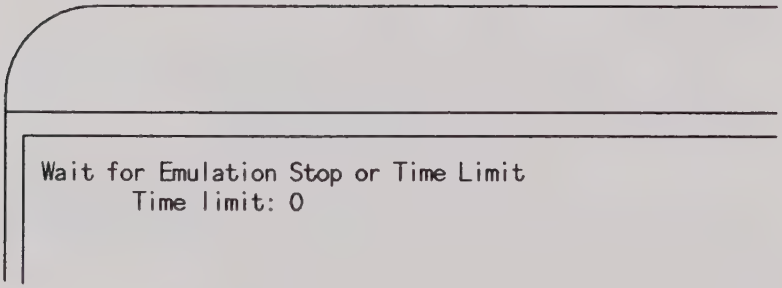
2) In direct command format, the previous command is entered as follows:

```
/tpr start=0 new=yes
```

PATH:

/
Target
Processor
Wait

OPTION WINDOW:



OPTION NAMES (direct command format):

time

DESCRIPTION:

After issuing a *wait* command, no further user commands are executed until the specified Time limit is expired or until the emulation run is halted. When Time limit is set to 0, which is the default value, no further user commands are executed until the emulation run is halted.

The default action of this command is to execute the previous *wait* command. The option for this command allows specification of the time limit in seconds. When the Debugger is invoked, the

time limit is set to 0.

When executing a *Processor Run* command from within a command file, a *Processor Wait* may be issued to make sure that the emulation run is halted before any other debugger commands are issued.

EXAMPLES:

1) After issuing the following command during an emulation run, no user commands will be executed for the first 45 seconds or until the emulation run is halted.

```
tpw

Wait for Emulation Stop or Time Limit
      Time limit: 45
```

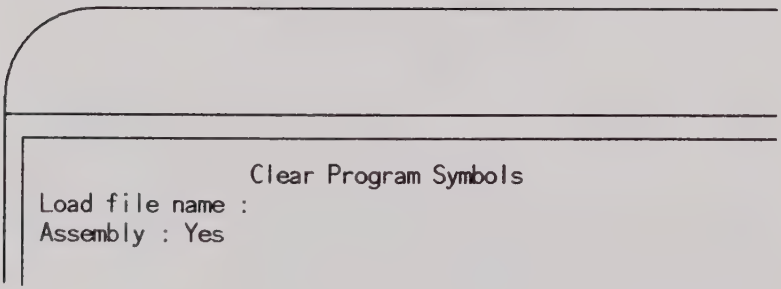
2) In direct command format the previous command is entered as follows:

```
/tpw time=45
```

PATH:

/
Objects
ProgSymb
Clear

OPTION WINDOW:



OPTION NAMES (direct command format):

file	asm

DESCRIPTION:

This command is used to delete symbolic information from the debugger table.

Load file name — Specifies the name of the file of which symbolic information must be deleted. Default is the name of the most recently loaded file.

Assembly Specifies that assembly level symbols have to be deleted.

EXAMPLES:

1) The following command deletes all symbolic information of the load file *a.out*:

```
opc

                    Clear Program Symbols
Load file name : a.out
Assembly : Yes
```

Enter Enter

PATH:

/
Quit

DESCRIPTION:

This command terminates the debugging session.

EXAMPLES:

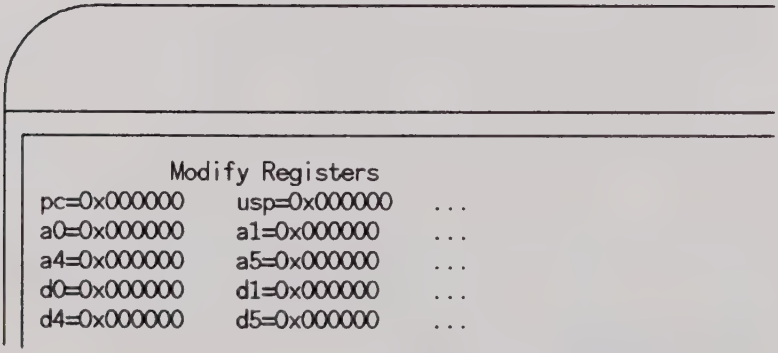
Debugging is curtailed when pressing:

q
Enter

PATH:

/
Target
Registers
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

a0-a6	pc	usp
d0-d7	sr	ssp

option names for the 68010 only:

vbr	dfc
sfc	fow

DESCRIPTION:

This command allows the user to alter the values of one or more registers. By using the arrow keys the register of which the value has to be altered can be selected. The value can be entered using the character keys.

EXAMPLES:

1) The content of the register *a4* is set to 0x7a:

```
trm

...
a0 = 0x000000  a1 = 0x000000    ...
a4 = 0x000000  ...
```

Press ↓ ↓ and type 0x7a.

```
...
a0 = 0x000000  a1 = 0x000000    ...
a4 = 0x7a          ...
```

Press Enter twice.

2) In direct command format this command is entered as follows:

```
/trm a4=0x7a
```

PATH:

/
Target
Registers
Show

DESCRIPTION:

This command displays the contents of the target microprocessors register. The following register abbreviations are used:

a0 - a6 address register a0 to a6

d0 - d7 data register d0 to d7

pc program counter

sr status register

usp user stack pointer

ssp supervisor stack pointer

The registers below are 68010 registers only.

vbr vector base register (max. value 0xffffff)

sfc source function code register (max. value 0x7)

dfc destination function code register (max. value 0x7)

fow format/offset word (max. value 0xffff)

EXAMPLES:

1) The content of the registers is shown:

```
/trs
Enter
```

```
pc = 0x00000000 ; usp = 0x00000000 ;  ...
a0 = 0x02016084 ;  a1 = 0x00000000 ;  ...
a4 = 0x0000fff0 ;  a5 = 0x00000000 ;  ...
d0 = 0x934a0812 ;  d1 = 0x00000000 ;  ...
...
```

PATH:

/
Services
Session
Display

DESCRIPTION:

This command displays the current logging state of the command logging file and the display logging file. When logging is enabled for a particular file, the log file name is also shown.

EXAMPLES:

1) The following command displays the current logging status of both log files:

```
ssd
Enter
```

```
Command output is being appended to log.cmd
Display output is being appended to log.cmd
```

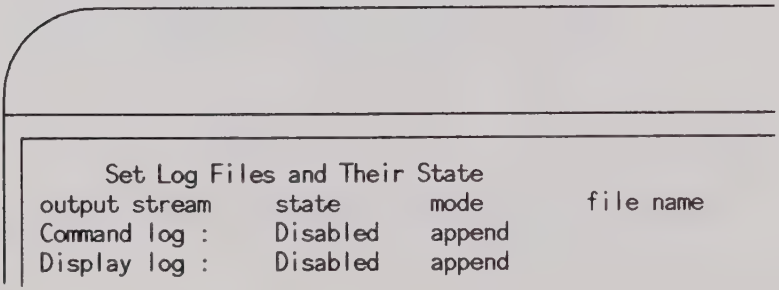
2) In direct command format the previous command is entered as follows:

```
ssd
```

PATH:

/
Services
Session
Modify

OPTION WINDOW:


Set Log Files and Their State
output stream state mode file name
Command log : Disabled append
Display log : Disabled append

OPTION NAMES (direct command format):

cmdlog	displog
cmdmode	dispmode
cmdfile	dispfile

DESCRIPTION:

This command allows the creation and closure of command and display log files.

State The state of a log file is either enabled or disabled. By default, it is set to disabled.

Mode When enabling a log file, the user can decide to append log data to the file or to overwrite the contents of the file. By default, mode is set to append.

File name Specifies the name of the log file. When only one file name is specified and both log files are enabled, then the commands entered and the command output are logged to this file. The default file name for both command and display logging is *debug.log*.

EXAMPLES:

1) The following command is entered to append all debugger output to a file *logfile1*:

```
'ssm
```

```
      Set Log Files and Their State
output stream     state      mode       file name
Command log :     Enabled    append     logfile1
Display log :     Enabled    append     logfile1
```

Enter Enter

```
Command output is being appended to logfile1
Display output is being appended to logfile1
```

2) In direct command format the previous command may be entered as follows:

```
'ssm cmdlog=enabled displog=enabled cmdfile=logfile1
```

PATH:

/
Services
Session
Resume

DESCRIPTION:

This command resumes logging to both the command log file and the display log file.

EXAMPLES:

1) Logging is resumed:

ssr
Enter

Note that no confirmation is given after entering this command.

PATH:

/
Services
Session
Suspend

DESCRIPTION:

This command suspends logging to both the command log file and the display log file.

EXAMPLES:

1) Logging is suspended:

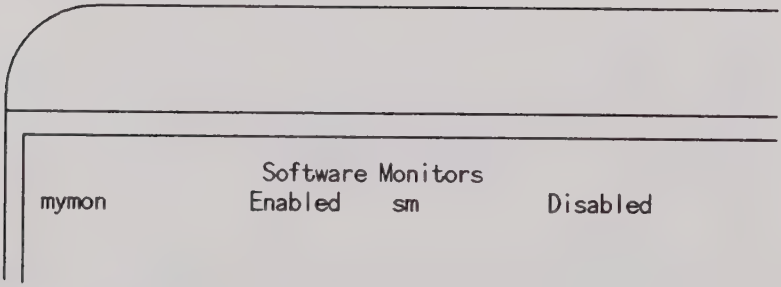
/sss
Enter

Note that no confirmation is given after entering this command.

PATH:

/
Debugger
SwMonitor
Disable

OPTION WINDOW:



OPTION NAMES (direct command format):

The user may specify one or more names of existing software monitors.

DESCRIPTION:

This command allows the user to enable or disable a software monitor. The + or − keys can be used to alter the status of a software monitor.

EXAMPLES:

1) All software monitors and their current status are displayed when issuing the following command:

```
dsd
```

```
                    Software Monitors
mymon          Enabled    sm          Disabled
```

The software monitor *mymon* may be disabled by pressing + Enter Enter .

```
Software Monitor disabled.
```

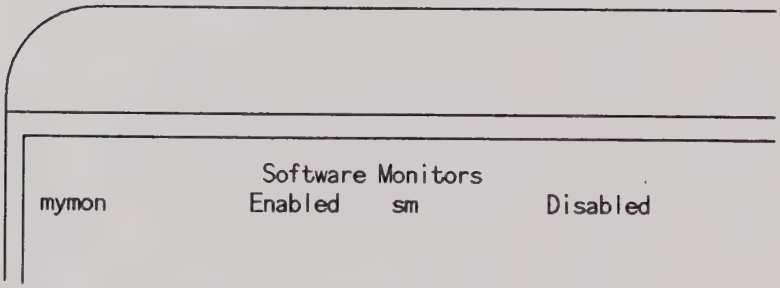
2) In direct command format the previous command may be entered as follows:

```
/dsd sm=disabled
```

PATH:

/
Debugger
SwMonitor
Enable

OPTION WINDOW:



OPTION NAMES (direct command format):

The user may specify one or more names of existing software monitors.

DESCRIPTION:

This command allows the user to enable or disable a software monitor. The + or − keys can be used to alter the status of a software monitor.

EXAMPLES:

1) All software monitors and their current status are displayed when issuing the following command:

```
dse
```

```
                    Software Monitors
mymon               Enabled     sm              Disabled
```

The software monitor *mymon* may be disabled by pressing + Enter Enter .

```
Software Monitor disabled.
```

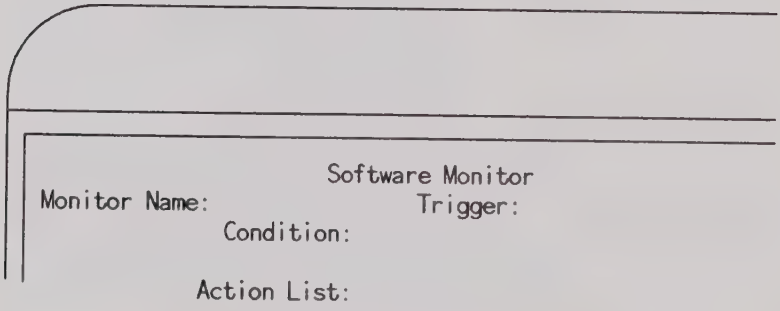
2) In direct command format the previous command may be entered as follows:

```
/dse sm=disabled
```

PATH:

/
Debugger
SwMonitor
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

name
trigger
cond

DESCRIPTION:

This command creates a software monitor, and automatically enables it.

The arguments have the following meanings:

Monitor Name the specified name must be unique and is the name assigned to the software monitor. The name entered may not exceed twenty characters.

Trigger either the + or − key may be used to specify one of the following triggers:

tr0 the monitor is activated when the conditions specified by the hardware monitor *tr0* are met. This is the default trigger.

tr1 the monitor is activated when the conditions specified by *tr1* are met.

qr0 the monitor is activated when the conditions specified by *qr0* are met.

qr1 the monitor is activated when the conditions specified by *qr1* are met.

rtec the monitor is activated when a break is generated by the real time event counting mechanism.

instr the monitor is activated each time an assembly instruction has been executed.

Action List consists of one or more debugger commands which specify the action(s) to be performed. The commands have to be entered on separate lines in direct command format.

Condition specifies the condition that must be matched before any action is taken.

EXAMPLES:

1) A software monitor called mymon is created:

```
dsm
```

```
                        Software Monitor
Monitor Name: mymon              Trigger: qr0
                   Condition:
a==b
                 Action List:
/trs
```

Enter Enter

```
Software Monitor created
```

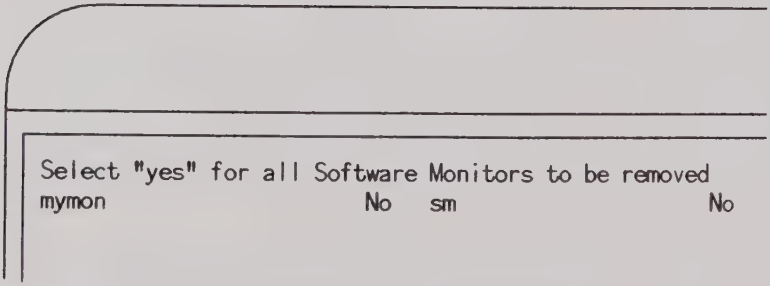
2) In direct command format this command may be entered as follows:

```
dsm name mymon trigger=qr0 cond=a= =b
        trs
        sae
```

PATH:

/
Debugger
SwMonitor
Remove

OPTION WINDOW:



OPTION NAMES (direct command format):

The user may specify one or more names of existing software monitors.

DESCRIPTION:

This command allows the deletion of one or more software monitors. In order to remove a software monitor, the value displayed next to the monitor has to be set to *Yes*. The + or − keys are used to select either of the values *Yes* or *No*.

EXAMPLES:

1) The software monitor *sm* is deleted:

dsr

```
Select "yes" for all Software Monitors to be removed
mymon                    No    sm                    No
```

Press → + Enter Enter .

```
Software Monitor removed
```

2) In direct command format the previous command may be entered as follows:

/dsr sm=Yes

PATH:

/
Debugger
SwMonitor
Show

DESCRIPTION:

This command displays the name, the status and the trigger of all software monitors.

EXAMPLES:

1) This command shows that two software monitors *sm* and *mymon* are currently defined.

/ds
Enter

```
Status of Software Monitors
mymon                enabled  tr0
sm                   enabled  qr1
```

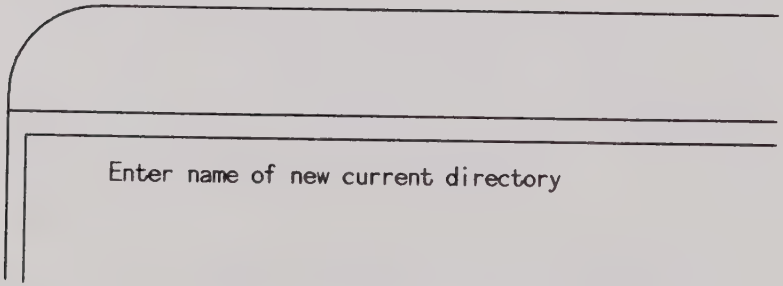
2) In direct command format the previous command is entered as follows:

```
dss
```

PATH:

/
Files
System
Chdir

OPTION WINDOW:



OPTION NAMES (direct command format):

dir

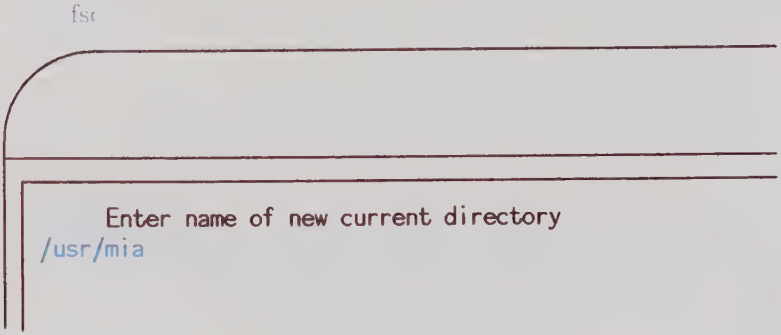
DESCRIPTION:

This command allows the user to change the current working directory, without having to leave the Debugger. The default action of this command is to revert to the original working directory from which the Debugger was invoked.

The option for this command is used to enter the path name of the new working directory.

EXAMPLES:

1) The current working directory is set to /usr/mia:



Press Enter twice to execute the command.

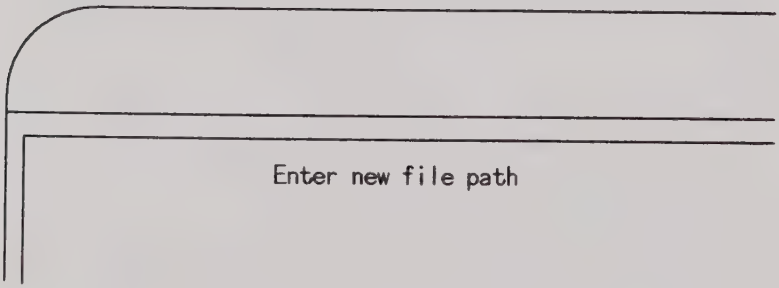
2) In direct command format, the previous command has to be entered as follows:

```
/fsc  dir=/usr/mia
```

PATH:

/
Files
System
Path

OPTION WINDOW:



OPTION NAMES (direct command format):

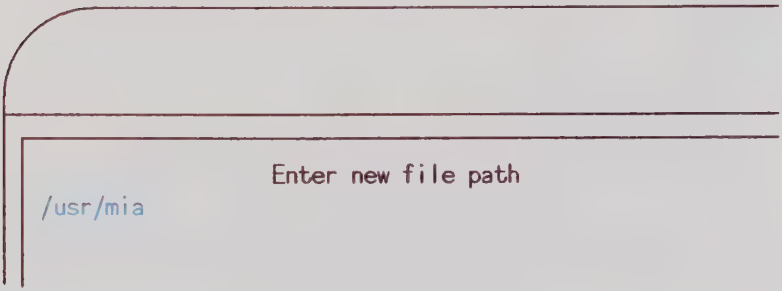
path

DESCRIPTION:

This command allows the user to change the default search path for the directory containing a file. The value of *Path* must be a string containing directory names, separated by a colon. The default *Path* is ".".

EXAMPLES:

1) The default search path is changed to /usr/mia:



Enter Enter

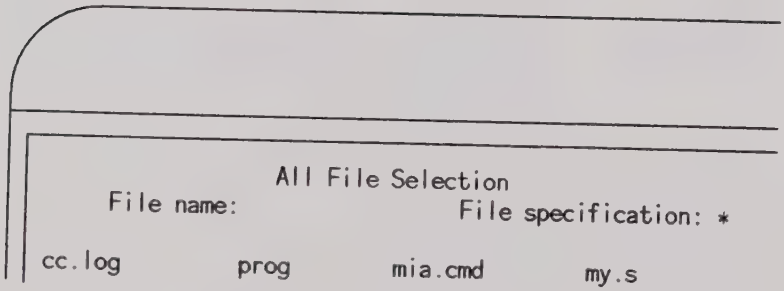
2) In direct command format, the previous command is entered as follows:

```
/fsp path=/usr/mia
```

PATH:

/
Files
System
Remove

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

This command removes any type of file. There is no default action for this command. As such, a pop-up window has to be displayed to select a file.

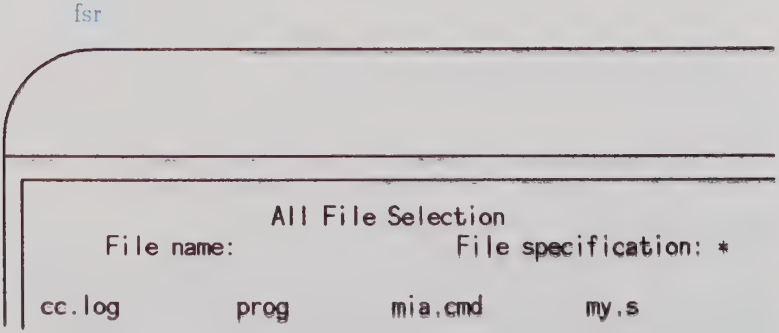
File name Specifies the name of the file to be removed. This name can be selected from the list of displayed files, which are the files present in the current working directory. This can be done by using the arrow keys to highlight the required file, and subsequently pressing Enter.

It is also possible to type the name of a file using the character keys.

File specification By default, this option is set to *, which means that any file may be removed. The user may alter the value of the file specification.

EXAMPLES:

1) The file *my.s* is removed.



Press ↓ End Enter Enter

2) In direct command format the previous command may be entered as follows:

```
/fsr file=my.s
```

PATH:

/
Debugger
Trace
Disable

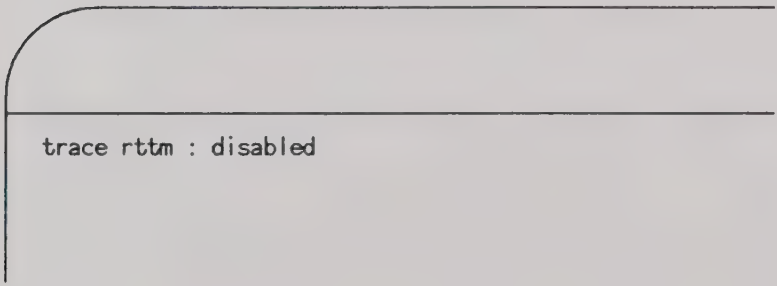
DESCRIPTION:

This command disables the real-time trace mechanism, without destroying the real-time memory contents.

EXAMPLES:

1) The real-time trace mechanism is disabled:

/dtd
Enter



PATH:

/
Debugger
Trace
Enable

DESCRIPTION:

This command enables the real-time trace mechanism. By default, the trace mechanism is enabled.

EXAMPLES:

1) The real-time trace mechanism is enabled:

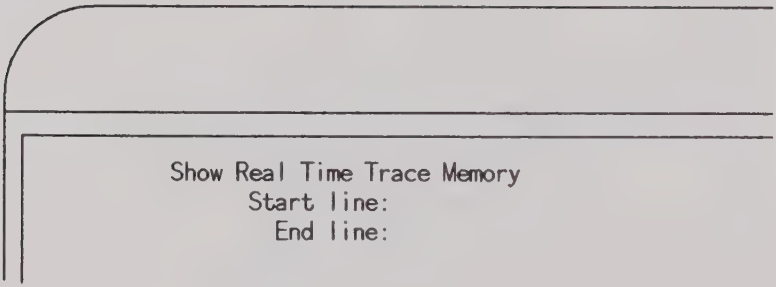
```
/dte
Enter
```

```
trace rttm : enabled
```

PATH:

/
Debugger
Trace
List

OPTION WINDOW:



OPTION NAMES (direct command format):

start
end

DESCRIPTION:

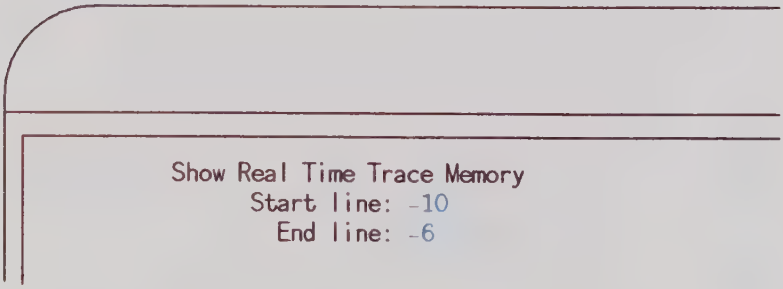
This command shows the contents of the real time trace memory.

The options *Start line* and *End line* allow the user to specify which entries in trace memory have to be displayed. By default, the first line of trace memory is displayed.

EXAMPLES:

1) The contents of line −10 to line −6 of the real time trace memory are displayed when entering the following command:

```
dtl
```



Enter Enter .

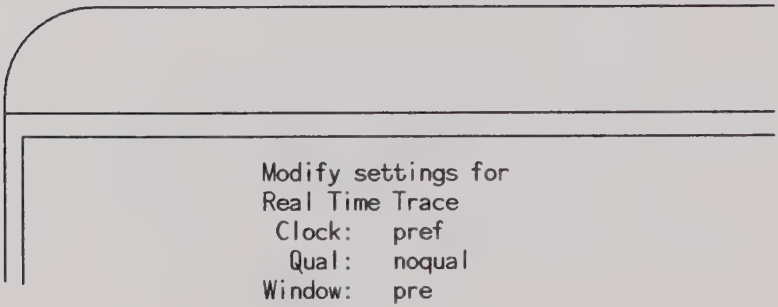
2) In direct command format the previous command is entered as follows.

```
/dtl start=-10 end=-6
```

PATH:

/
Debugger
Trace
Modify

OPTION WINDOW:



OPTION NAMES (direct command format):

clock
qual
window

DESCRIPTION:

This command can be used to alter the settings of the *trace-clock*, *trace-qualifier* and *trace-window* parameters.

Clock Specifies the location or source of the clock which will be used to provide triggering for the trace sampling logic. The following sources may be selected by means of the + or − keys:

pref user program or supervisor program accesses (program references).

mc machine cycles.

pod07lh clock pulse derived from positive edge of signal at pin 7 of pod 0.

pod07hl clock pulse derived from negative edge of signal at pin 7 of pod 0.

Qual Specifies the qualification, if any, which will be placed upon entering any data into real-time trace memory. The following qualifications can be selected by using the + or − key:

noqual unconditional tracing

qr0 trace entries are qualified by the value of the hardware monitor qr0.

qr1 trace entries are qualified by the value of the hardware monitor qr1.

qr trace entries are qualified by either of the hardware monitors qr0 or qr1.

NOTE

qr, *qr0* and *qr1* can only be useful as trace qualifiers if the action of the hardware monitors *qr0* and/or *qr1* is set to *pulse*. A warning is displayed when this is not the case.

Window Specifies the location at the trace window around the match point. The following trace mechanisms can be selected using the + or − key:

pre	the trace memory contains the last entries (up to 255) made up to the point at which the match point was detected.
centre	trace memory contains the last entries (up to 127) before the match point and the first (up to 127) entries after the match point.
post	trace memory contains the first entries (up to 255) after the match point was detected.

EXAMPLES:

1) The *trace-qualifier* parameter is set to *qr1*:

```
/dtm

Modify settings for
Real Time Trace
 Clock:   pref
  Qual:   noqual
Window:   pre
```

Press the keys ↓ and − to set Qual to *qr1*.

```
Modify settings for
Real Time Trace
 Clock:   pref
  Qual:   qr1
Window:   pre
```

Press [Enter] [Enter] .

```
trace rttm entries : -9 .. 0
trace rttm : enabled
trace rttm = pref qr1 pre
```

2) In direct command format the previous command may be entered as follows:

```
/dtm qual=qr1
```

PATH:

/
Debugger
Trace
Show

DESCRIPTION:

This command displays the current settings of the real-time trace mechanism. The first line of screen output shows whether the trace mechanism is enabled or disabled. The second line shows the current settings of the parameters *trace-clock*, *trace-qualifier* and *trace-window*.

EXAMPLES:

1) The current settings of the trace memory are shown:

```
/dt
Enter
```

```
trace rttm : empty
trace rttm : enabled
trace rttm = mc qr centre
```

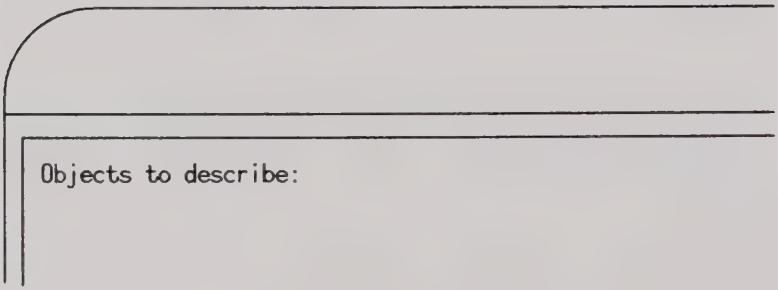
2) In direct command format the previous command is entered as follows:

```
dts
```

PATH:

/
Objects
Variables
Describe

OPTION WINDOW:



OPTION NAMES (direct command format):

names

DESCRIPTION:

This command displays the type of a program object. The type may be preceded by *file* :: where *file* is the source file name in which the object is defined. The type is always *assembler label.*

The default action of this command is to show the type of the last program object(s) referenced by any command of this group.

The options for this command allow the user to enter one or more program objects of which the types have to be displayed.

EXAMPLES:

1) The following command displays the source-path of the program object *msg* and shows that msg is an assembler label:

```
ovd
```

```
Objects to describe:
msg
```

Enter Enter

```
"main.s" ::   msg : assembler label;
```

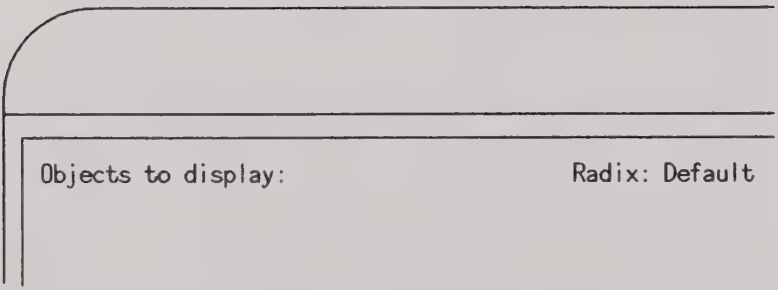
2) In direct command format the previous command is entered as follows:

```
/ovd names=msg
```

PATH:

/
Objects
Variables
Examine

OPTION WINDOW:



OPTION NAMES (direct command format):

names

DESCRIPTION:

This command displays the current value of the specified program objects. The default action of this command is to show the value of the last program object(s) referenced by any command of this group.

The options for this command permit the user to enter one or more program objects of which the values have to be displayed. The Radix option specifies the format in which the object value is displayed. It must be one of the following:

default the Debugger displays the value of an object in the most appropriate radix (hexadecimal or decimal). This is the default value.

hex hexadecimal format

decimal decimal format

octal octal format

binary binary format

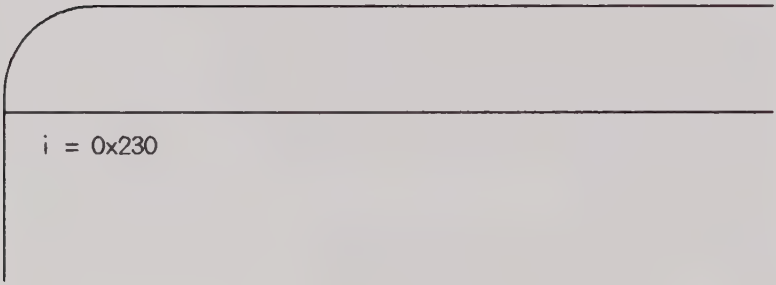
ascii ascii format

assembly returns the last label before the specified address and the offset of the address from that label. If no label is found, a message will be displayed.

EXAMPLES:

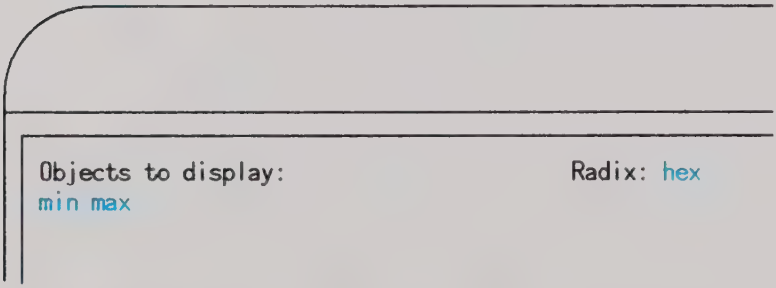
1) The value of the program object *i*, which is the last referenced program object by any command of the Variables group, is displayed:

```
/ove
Enter
```

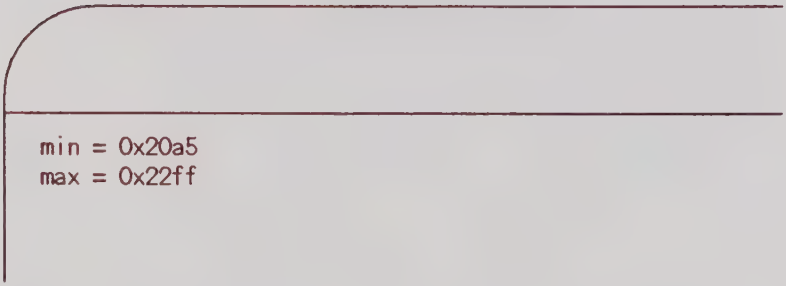


2) The addresses of the labels *min* and *max* are displayed:

/ove



Enter Enter



```
min = 0x20a5
max = 0x22ff
```

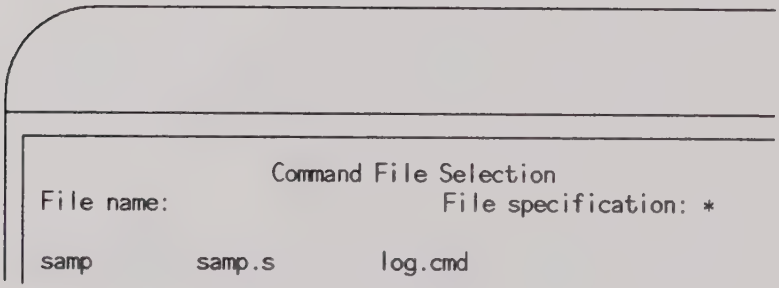
3) In direct command format this command is entered as follows:

```
/ove radix=hex names=min max
```

PATH:

/
Files
View
All

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

The default action for this command is to display the current default file for the *Files All* group.

The option for this command allows the user to specify a file to be displayed.

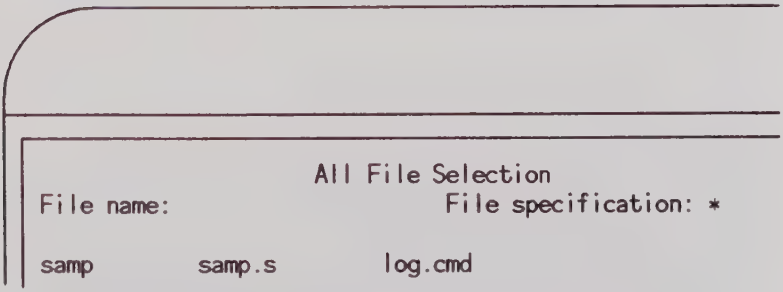
File name Specifies the name of the file to be viewed. The file name can be selected from the list of displayed files, which are the files present in the current working directory. This can be done by using the arrow keys to highlight a file name and by subsequently pressing Enter . The user can also type the file name using the character keys.

File specification Specifies that a file name, no matter what suffix, has to be selected. The user may alter the file specification.

EXAMPLES:

1) The following command allows the user to select a file to be viewed.

```
fva
```



The file *samp.s* is displayed when pressing ↓ → Enter Enter

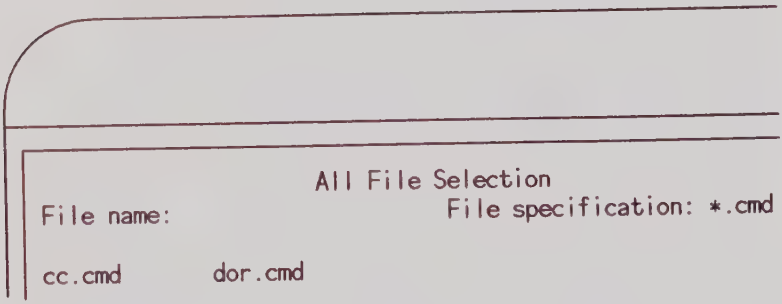
2) In direct command format the previous command is entered as follows:

```
/fva file=samp.s
```

PATH:

/
Files
View
Command

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

The default action for this command is to display the current default file for the *Files All* group.

The option for this command allows the user to specify a command file which is to be displayed.

File name Specifies the name of the command file to be viewed. The file name can be selected from the list of displayed files, which are the files present in the current working directory and ending *.cmd*. This can be done by using the arrow keys to highlight a file name and subsequently press Enter . The user can also type the command file name using the character keys.

File specification Specifies that a command file name, ending *.cmd*, has to be selected. The user may alter the file specification.

EXAMPLES:

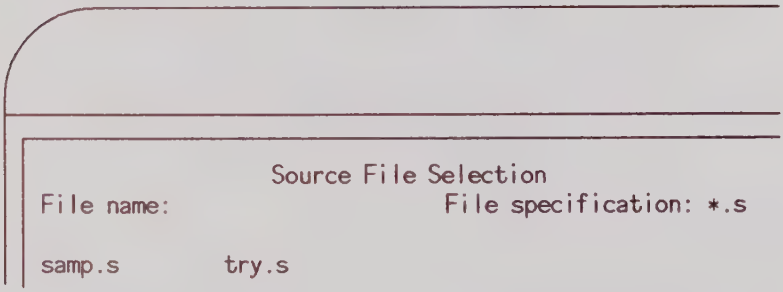
1) The following command displays the current default file for the command files group.

```
/fv
Enter
```

PATH:

/
Files
View
Source

OPTION WINDOW:



OPTION NAMES (direct command format):

file spec

DESCRIPTION:

The default action for this command is to display the current default file for the *Files Source* group.

The option for this command allows the user to specify a source file which is to be displayed.

File name Specifies the name of the source file to be viewed. The file name can be selected from the list of displayed files, which are the files present in the current working directory and ending *.s*. This can be done by using the arrow keys to highlight a file name and subsequently press Enter . The user can also type the command file name using the character keys.

File specification Specifies that a source file name, ending *.s*, has to be selected. The file specification may be set to any wild card specification.

EXAMPLES:

1) The following command displays the current default file for the command files group.

```
/fv
→
Enter
```

PATH:

/
Services
Window
Cycle

FUNCTION KEYS:

Cycle Wnd

DESCRIPTION:

The default window is cycled around the display windows currently present on the screen.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and three display windows. Window 1 is the default window.

   When issuing the command swc twice, Window 3 becomes the default window.

PATH:

/
Services
Window
Delete

FUNCTION KEYS:

Del Wnd

DESCRIPTION:

The *Delete Window* command deletes the default window.

The screen space freed by deleting a window is allocated to one of the remaining windows. It is not possible to delete the last display window from the screen.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and three display windows: Window 1 is the default window.

   If Window 2 has to be deleted, first press /swc and Enter to make Window 2 the default window. Subsequently, enter the command /swd to delete Window 2.

PATH:

/
Services
Window
Make

FUNCTION KEYS:

Make Wnd

DESCRIPTION:

This command creates a new window display on the screen. This is done by splitting the default window into two. The top half remains in charge of the command currently active within the default window and, as such, its content does not change. The bottom half belongs to the newly created window and is initially empty.

A maximum of four display windows can be on the screen. The *Make* command will fail if the user tries to exceed this number.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu window and one display window.

   A new window is created by pressing:

   /swm
   Enter

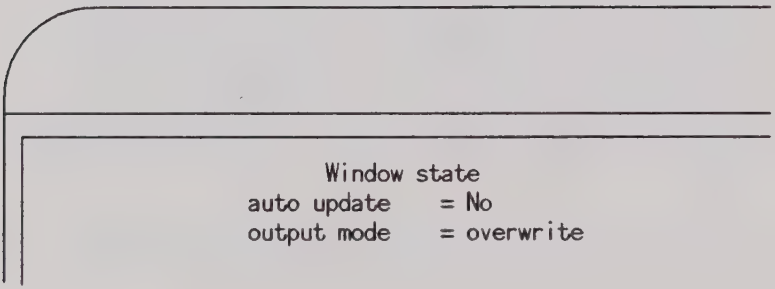
2) Assume that the basic screen currently consists of a menu window and 2 display windows. Window 1 is the default window.

We want to create a new window display at the bottom of the screen. In other words, we want Window 2 to be split into two. Therefore, first enter the command /swc (explained in the *Cycle Window* command) to make Window 2 the default window. Subsequently, the command /swm has to be issued to create a new window.

PATH:

/
Services
Window
State

OPTION WINDOW:



OPTION NAMES (direct command format):

auto mode

FUNCTION KEYS:

Append

Update

DESCRIPTION:

The default display window can be updated each time an emulation run is halted. In addition, the way debugger

commands are added to the default display window can be set. There are two possible modes: *append to window* and *overwrite window*. The latter is the default mode.

In the *overwrite window* mode, the output of a command issued deletes any previous display in the window. The advantage of this mode is that only the amount of memory required to display a single command in the window is used. In *append* mode, command output is appended to the display currently in the window. Therefore it is possible to display several commands in a single window. In this case, a circular fixed-sized buffer is used to hold the command output. The oldest command output will be lost first from the buffer. The disadvantage of this mode is that for each window, there must be a large fixed size memory buffer.

Setting the state of a window can be done by issuing a /sws command, or using the function keys [Append] and [Update].

- After issuing the command sws, the following options may be set:

  auto update Determines whether or not the default display window is updated each time an emulation run is halted. In direct command mode this is set by *auto=No* for no update or *auto= Yes* for update.

  output mode Determines the mode. In direct command mode this is set by *mode=append* or *mode=overwrite.*

- The function key [Append] is used to update the default display window each time an emulation run is halted. The function key [Update] is used to alter the current mode.

EXAMPLES:

1) Assume that the basic screen currently consists of the menu and two display windows. Window 1 is the default window.

   By default, the window mode of Window 2 is set to *overwrite window*. To set the mode of this window to *append* mode, first press [Cycle Wnd] to make Window 2 the default display window. Pressing [Update] now changes the window mode of Window 2 to *append* mode.

   The mode of Window 2 can be reset to overwrite mode by pressing [Update] .

CONTENTS

		Page
5	DEBUGGING AN ASSEMBLY PROGRAM	5-3
5.1	Overview	5-3
5.2	System Requirements	5-3
5.3	Assembling the Sample Program	5-4
5.4	Invoking the Debugger	5-5
5.5	The Menu Window	5-6
5.6	Logging	5-9
5.7	Mapping the Emulation Memory	5-12
5.8	Loading and Running the Program	5-18
5.9	Examining the Program	5-21
5.10	Exiting from the Debugger and Patching	5-30
5.11	Program Listing	5-32

CHAPTER 5

DEBUGGING AN ASSEMBLY PROGRAM

5.1 Overview

This chapter illustrates the use of the PMDS Assembly Debugger for the debugging of assembly programs. The first few sections provide a description of the hardware components required and the features of the Debugger of which the user must be aware in order to start. The main part of the chapter explains how to debug an example assembly program which takes the new user through the most commonly used debugger commands. The usage of keyboard keys and different windows is explained. The sample program used contains a bug which is found during the debugging session.

As well as loading the software, the installation procedure also loads the example program used in this chapter. The program can be found in the directory */usr/examples/debug/68000.*

5.2 System Requirements

Before starting with the Debugger, check that the following components are installed:

a) The AT Extension Box (PMDS-III)
b) Emulation Memory (minimum 32 Kbyte DRAM)
c) 68000/68010 MAB and Processor
d) 68000/68010 Debugger Software
e) 68000/68010 Assembler

If a) or b) above is not present, it should be installed according to the procedure described in the *PMDS-III Installation Guide* or the *PMDS-II Debugger Manual.* If c) or d) above is not present, it should be installed according to the installation procedure described in the *Installation Manual.* If e) above is not present, it should be installed according to the installation procedure described in the *PMDS-III Cross-Assembler 68000 Manual* or in the *PMDS-II Cross-Assembler 68000 Bulletin.*

The MAB must be set to simulation mode. This can be done by connecting the Probe to the simulation socket on the MAB.

5.3 Assembling the Sample Program

The program *optext.s* is the same example as used and explained in Chapter 2 of the *PMDS-III Cross-Assembler 68000 Manual* or the *PMDS-II 68000 Cross-Assembler Tutorial.* The program is split across a *main* module and a *_write* routine.

Briefly, the sample program outputs the string *'PCP: 68000'* via an ACIA. The program begins by loading three parameters onto the stack. It then branches to a *_write* routine which first unloads these parameters. After testing for the readiness of the ACIA, the routine outputs the first character of the string and tests for any more characters. When all the characters have been output, the program returns from the subroutine, cleans the stack and stops.

As mentioned above, the sample program *optext.s* is contained in the directory */usr/examples/debug/68000*. It has to be assembled and linked into an *out.h* format file that the Debugger requires.

```
68000 optext.s -o opprog
```

If the program assembles and links successfully, the result will be in the file *opprog* in the current directory.

Names of sections, labels and constants which are declared by the user in the source program, are called symbols. For instance, *NOCHARS*, *msg* and *_write* are symbols defined in *optext.s*.

5.4 Invoking the Debugger

The Debugger may be invoked as follows:

```
Adeb68000
```

The Debugger works in screen oriented mode. This means that the screen is divided into several rectangular areas or windows through which data can be viewed.

When invoking the Debugger, the following information appears on the screen:

```
Philips PMDS 68000/10 Asm Debugger
Config    Debugger Target  Objects   Files  Services...
MemoryMap Clock    Pod     Interrupt

(C) 1987 Philips Export BV. All rights reserved.
PF8219M Debug processor asm-68010/68000   Release 2.0

 Selected debug unit: d0

 Connected MAB: PM8329/PM8328   Processor: 68000

 environment variable EDITOR defaulted to "vi"
 environment variable VIEW defaulted to "more"
```

The upper window or menu window consists of two lines, containing command names which can be selected to execute debugger commands during a session. The lower window, which is a display window, contains a message indicating the version number of the software and the debug unit and target microprocessor selected.

The Debugger contains a help facility which provides information on the debugger command currently selected. This help facility can be invoked at any time during a session by pressing the Help key.

5.5 The Menu Window

Before a command can be executed, it has to be selected from the menu window. The menu window is organised as a tree structure. The top level contains the names of command groups. The second and third level contain command names. An

overview of the structure of the menu display is given on the next page.

When entering the Debugger, the top level and second level of the menu window are displayed. The cursor box, indicating the command which may be currently selected, is highlighted on the first command of the top level.

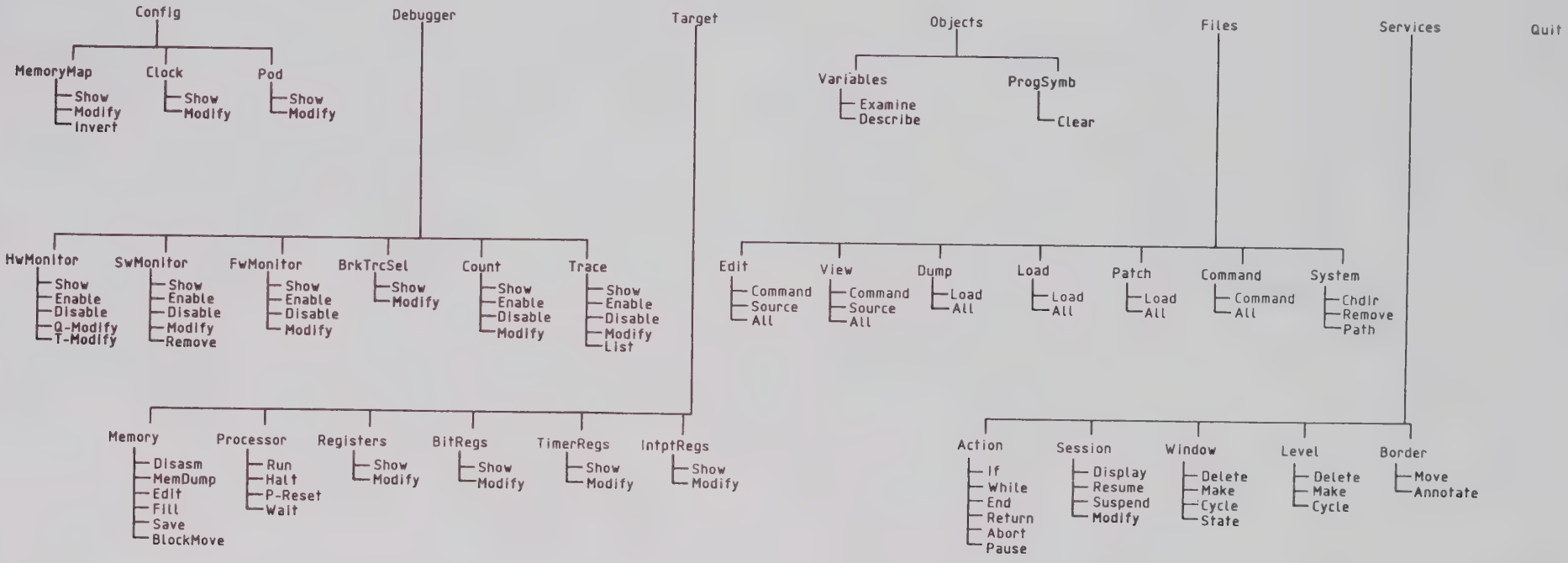
Cursor keys can be used to move across commands in the menu tree:

- when pressing cursor right, the cursor moves to the next command of the current level.

- when pressing cursor left, the cursor moves to the previous command of the current level.

- when pressing cursor down, the cursor moves to the first command of the next level.

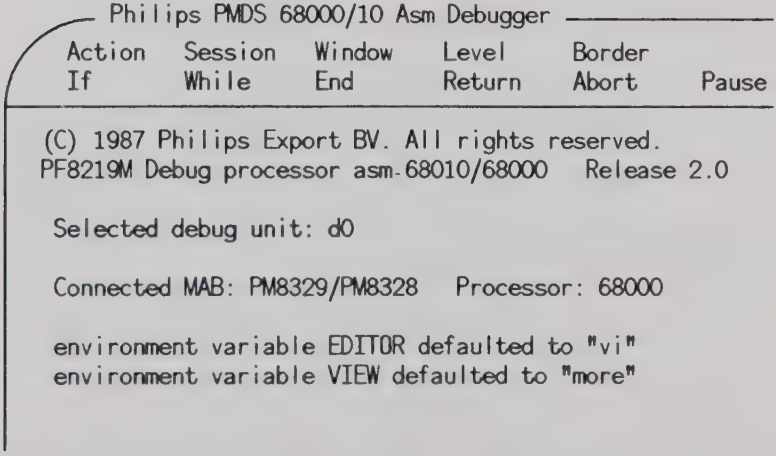
- when pressing cursor up, the cursor moves up one level in the tree.


Config
MemoryMap
Show
Modify
Invert
Clock
Show
Modify
Pod
Show
Modify
Debugger
HwMonitor
Show
Enable
Disable
Q-Modify
T-Modify
SwMonitor
Show
Enable
Disable
Modify
Remove
FwMonitor
Show
Enable
Disable
Modify
BrkTrcSel
Show
Modify
Count
Show
Enable
Disable
Modify
Trace
Show
Enable
Disable
Modify
List
Target
Memory
Disasm
MemDump
Edit
Fill
Save
BlockMove
Processor
Run
Halt
P-Reset
Wait
Registers
Show
Modify
BitRegs
Show
Modify
TimerRegs
Show
Modify
IntptRegs
Show
Modify
Objects
Variables
Examine
Describe
ProgSymb
Clear
Files
Edit
Command
Source
All
View
Command
Source
All
Dump
Load
All
Load
Load
All
Patch
Load
All
Command
Command
All
System
Chdir
Remove
Path
Services
Action
If
While
End
Return
Abort
Pause
Session
Display
Resume
Suspend
Modify
Window
Delete
Make
Cycle
State
Level
Delete
Make
Cycle
Border
Move
Annotate
Quit
MA11822

5.6 Logging

A command log file, recording all commands issued, and a display log file, recording all command output, may be opened at any time during a debugging session. The *Session Modify* command is provided to open and close log files. As this command belongs to the *Services* command group, first press cursor right five times to move the cursor to *Services*.

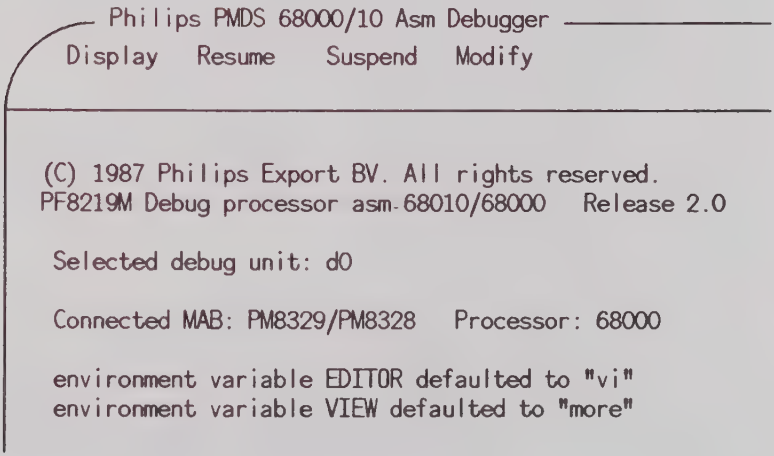
When pressing cursor down to select *Services*, the top line of the menu window is replaced by the next level showing all commands belonging to the *Services* group, and the cursor is positioned on the *Action* command.


Philips PMDS 68000/10 Asm Debugger
Action Session Window Level Border
If While End Return Abort Pause
(C) 1987 Philips Export BV. All rights reserved.
PF8219M Debug processor asm-68010/68000 Release 2.0
Selected debug unit: d0
Connected MAB: PM8329/PM8328 Processor: 68000
environment variable EDITOR defaulted to "vi"
environment variable VIEW defaulted to "more"

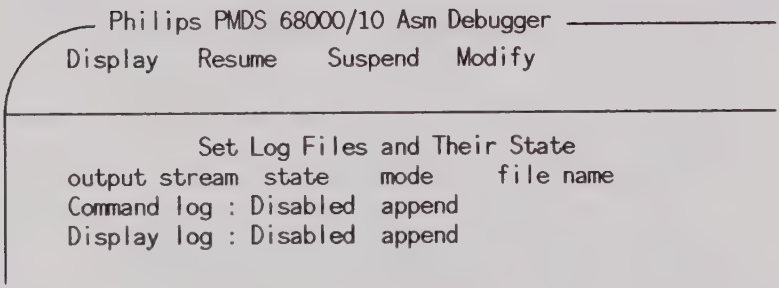
The *Services* command could also have been selected by pressing the character *s*. In general, commands displayed in the menu window may be selected by pressing their initial letter. This is a faster way than using cursor keys and therefore, we will often

prefer this method to select commands further on in the tutorial.

The *Session* command can now be selected by pressing the character *s*. As a result, the cursor is positioned on the first subcommand of *Session*.

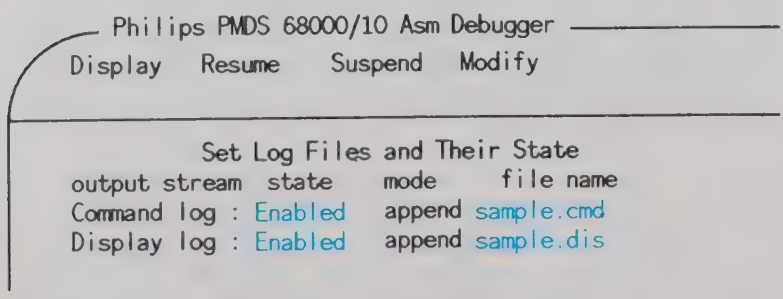

Philips PMDS 68000/10 Asm Debugger
Display Resume Suspend Modify
(C) 1987 Philips Export BV. All rights reserved.
PF8219M Debug processor asm-68010/68000 Release 2.0
Selected debug unit: d0
Connected MAB: PM8329/PM8328 Processor: 68000
environment variable EDITOR defaulted to "vi"
environment variable VIEW defaulted to "more"

Pressing the letter *m* selects the *Modify* command and displays the option window for this command.


Philips PMDS 68000/10 Asm Debugger
Display Resume Suspend Modify
Set Log Files and Their State
output stream state mode file name
Command log : Disabled append
Display log : Disabled append

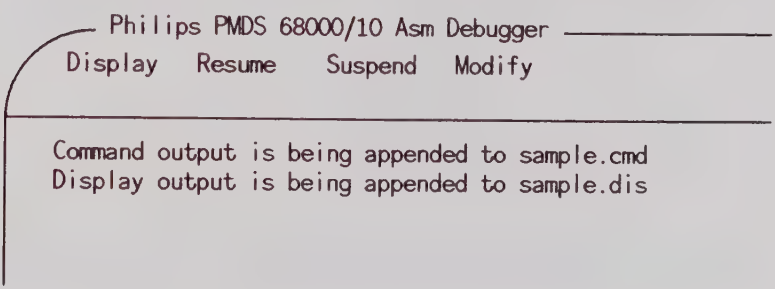
Note that at any time, an option window may be left by pressing Esc . The previous option values will be restored. The option window may be displayed again by pressing cursor down.

The cursor is now highlighted on Command's log state, which is set to *Disabled* by default. As a command log file has to be opened, we set its state to *Enabled* by pressing either of the + or − keys located on the keyboard. The file name to which all issued commands are logged needs to be specified. This is done by pressing → twice to preselect the option *file name*, and entering a file name, say *sample.cmd*. After pressing ↓, the display log file name is set in the same way to *sample.dis*. Press Enter , ↓ and + to enable display logging.


Philips PMDS 68000/10 Asm Debugger
Display Resume Suspend Modify
Set Log Files and Their State
output stream state mode file name
Command log : Enabled append sample.cmd
Display log : Enabled append sample.dis

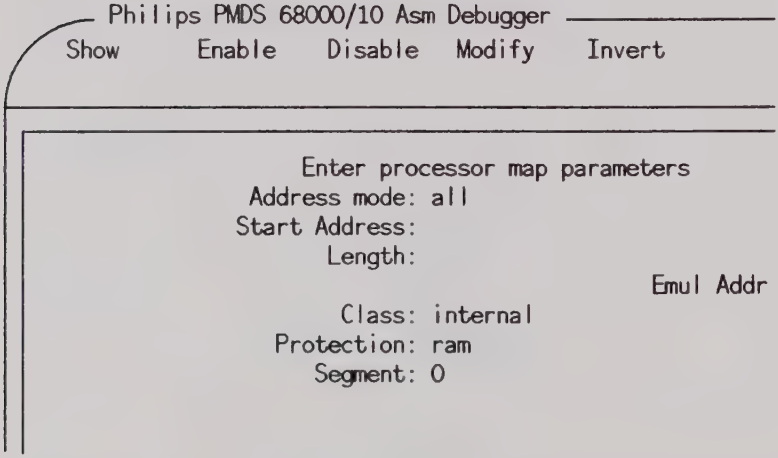
When pressing Enter once, the specified options are selected.

When pressing Enter a second time, the *Session Modify* command is executed and the following message is displayed:

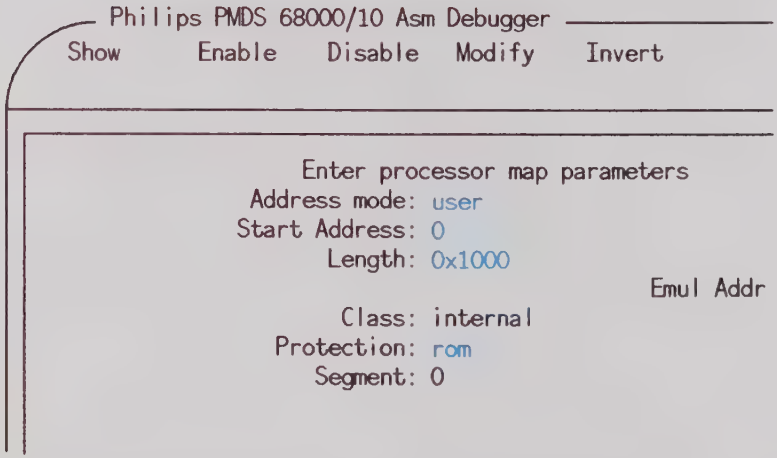

Philips PMDS 68000/10 Asm Debugger
Display Resume Suspend Modify
Command output is being appended to sample.cmd
Display output is being appended to sample.dis

5.7 Mapping the Emulation Memory

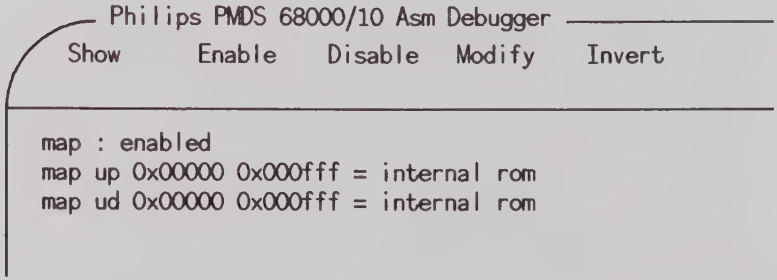
The first task is to set the emulation memory for the program, which sets the stack at address 0x1ffa and is designed for an ACIA being placed at address 0x1ffc. The program *optext.s* is divided into several sections. The main program starts at address 0x30 and will be executed in program mode. In order to simulate the use of a 32 Kbits EPROM, the first four Kbytes of emulation memory are mapped as ROM. This is done by issuing the /cmm (*Config MemoryMap Modify*) command. The / reinstates the top level of the menu. *c* selects *Config*, the first *m* selects *MemoryMap*, the second *m* selects *Modify* and displays the option window below:



When pressing the cycle key + once, Address mode is set to *user* meaning that the area is mapped to both 'user program' and 'user data'. The cursor down key may be used to move to the following options. The value of Start Address and Length are set to 0 and 0x1000 respectively. The argument *internal* indicates that internal emulation memory in the system is used. Protection is set to *rom* by pressing the cycle key +.

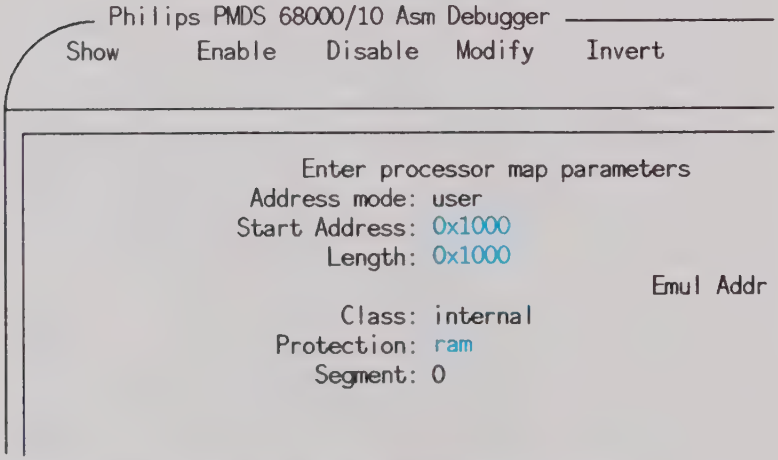


The command is executed when pressing Enter twice. The system responds immediately by displaying the area mapped:

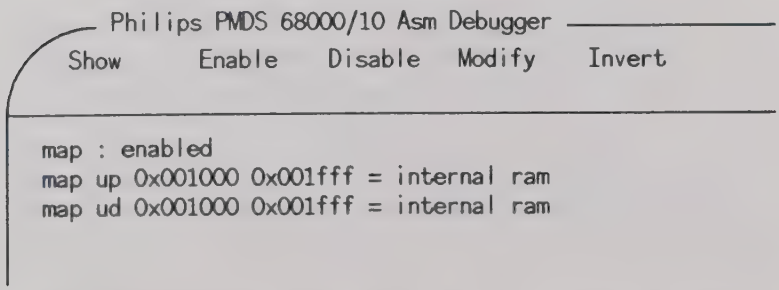

Philips PMDS 68000/10 Asm Debugger
Show Enable Disable Modify Invert
map : enabled
map up 0x00000 0x000fff = internal rom
map ud 0x00000 0x000fff = internal rom

We now have to map an area in which the stack is to reside and define the access to the ACIA. The area reserved for the stack is between 0x1000 and 0x1fff while the ACIA is located at address 0x1ffc. As it must be possible to read from and write to stack addresses and the ACIA, we will map 0x1000 to 0x1fff as RAM.

Press cursor down to display the option window of the *MemoryMap Modify* command, and set the options to the values shown below.



The command is executed when pressing Enter twice.


Philips PMDS 68000/10 Asm Debugger
Show Enable Disable Modify Invert
map : enabled
map up 0x001000 0x001fff = internal ram
map ud 0x001000 0x001fff = internal ram

Finally, the supervisor area located at 0 -0x2f should be mapped for the reset vectors which are located in the section *vectors* in the sample program. For this we need read-write-memory, so

Protection is set to *ram*. Both supervisor data and program memory have to be mapped. Therefore, Address Mode is set to *supervisor*. This can be done by pressing cursor down and setting the options to values shown below:

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Modify    Invert

                Enter processor map parameters
              Address mode: supervisor
             Start Address: 0
                    Length: 0x30
                                        Emul Addr
                     Class: internal
                Protection: ram
                   Segment: 0
```

The command is executed when pressing Enter twice.

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Modify    Invert

map : enabled
map sp 0x000000 0x000fff = internal ram
map sd 0x000000 0x000fff = internal ram
```

It can be checked that the system is mapped correctly by issuing a *Config MemoryMap Show* command. This can be done by first

pressing /cms, which preselects the *Show* command and displays the following option window:

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Modify    Invert

      Select the sections of the memory map to show
Long : No                        Address Mode : all
Start Address :
  End Address :
```

Start Address is set to 0, and End Address is set to 0x1fff:

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Modify    Invert

      Select the sections of the memory map to show
Long : No                        Address Mode : all
Start Address : 0
  End Address : 0x1fff
```

After pressing Enter twice, the following output is displayed on the screen:

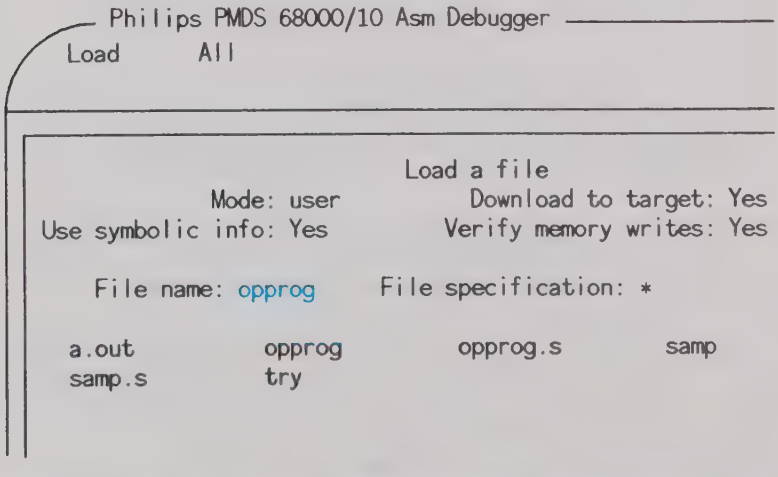
```
Philips PMDS 68000/10 Asm Debugger
  Show      Enable    Disable   Modify    Invert

map : enabled
map up 0x000000 0x000fff =  internal rom
map up 0x001000 0x001fff =  internal ram
map ud 0x000000 0x000fff =  internal rom
map ud 0x001000 0x001fff =  internal ram
map sp 0x000000 0x000fff =  internal ram
map sp 0x001000 0x001fff =  target   guarded
map sd 0x000000 0x000fff =  internal ram
map sd 0x001000 0x001fff =  target   guarded
```

This shows that the user program (*up*) area is set for read-only-memory in the range 0 to 0xfff, and read-write memory in the range 0x1000 to 0x1fff. The user data (*ud*) memory is set in a similar way. The supervisor area of memory, *sp* and *sd*, is set from 0 to 0xfff as read-write memory. The range 0x1000 to 0x1fff is set to guarded.

5.8 Loading and Running the Program

The program *opprog* may now be loaded by entering a /fll (*Files Load Load*) command. The option window below will be displayed. The option Mode specifies the mode in which the specified file is loaded. Use symbolic info which is set to Yes by default specifies that all symbols defined by the user in the sample program are loaded into the symbol table of the Debugger. Download to target which is set to Yes by default specifies that object code is loaded into emulation memory. Verify memory writes which is set to Yes by default specifies

that all loaded code is verified when being written to memory. The name of the file to be loaded needs to be specified. It can be selected from the list of files displayed, which includes all files in the current working directory, or it can be entered by the user. If positioned in a list of files, the user can return to the option fields by pressing Esc . As the file *opprog* is present in the list, the cursor keys may be used to move the cursor to *opprog*. The file may now be selected by pressing Enter .



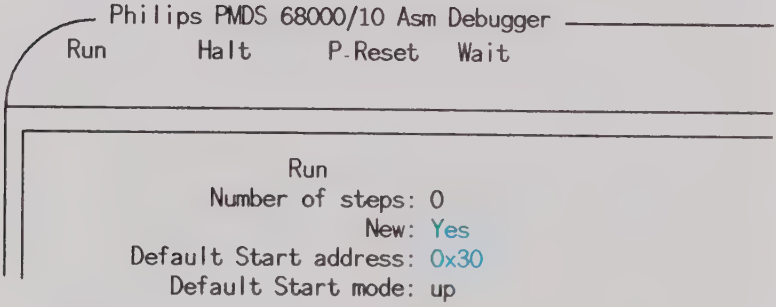
Press Enter to execute the command. While loading, the system displays an overview of all the loaded sections on the screen, as follows:

```
Philips PMDS 68000/10 Asm Debugger
Load      All

Reading symbols from load file "opprog"
Loading section  0 (vectors) : address = up 0x000000,...
Loading section  1 (program) : address = up 0x000030,...
Loading section  2 (.text) : address = up 0x00004e, ...
Loading section  3 (.data) : address = up 0x00009a, ...
Loading section  4 (.bss) : address = up 0x001000, ...
```

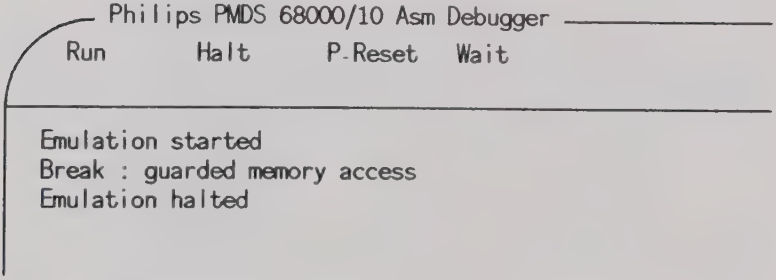
Now that we have opened a log file, mapped memory and loaded our program, we can run the program to see if it works. By typing /tpr, the *Target Processor Run* command is selected.

```
Philips PMDS 68000/10 Asm Debugger
Run       Halt      P-Reset   Wait

                    Run
          Number of steps: 0
                      New: No
  Default Start address: 0
     Default Start mode: up
```

When the New option is set to *Yes*, which can be done by pressing a cycle key, program execution starts at the address specified by Default Start address. Our program starts at address 0x30, so this address must be assigned to Default Start address. The Default Start mode is *up*.



Press Enter twice to start an emulation run. After some time the emulation run will be halted due to the error which is present and the following will be displayed:



This tells us that the program is trying to access data in an unmapped part of memory. However, further investigation will be necessary to determine why.

5.9 Examining the Program

The Real Time Trace Memory (*rttm*) stores history steps which contain microprocessor bus information. By default all

information is stored in trace memory. In order to examine the last 18 entries in the trace memory prior to the point that the emulation run stopped, enter a /dtl (*Debugger Trace List*) command and assign the values −17 and 0 to the options Start Line and End Line.

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Modify    List

          Show Real Time Trace Memory
        Start Line: -17
          End line: 0
```

After pressing Enter twice, history information showing a line number, an address, a disassembly of the program and the bus control mode is displayed:

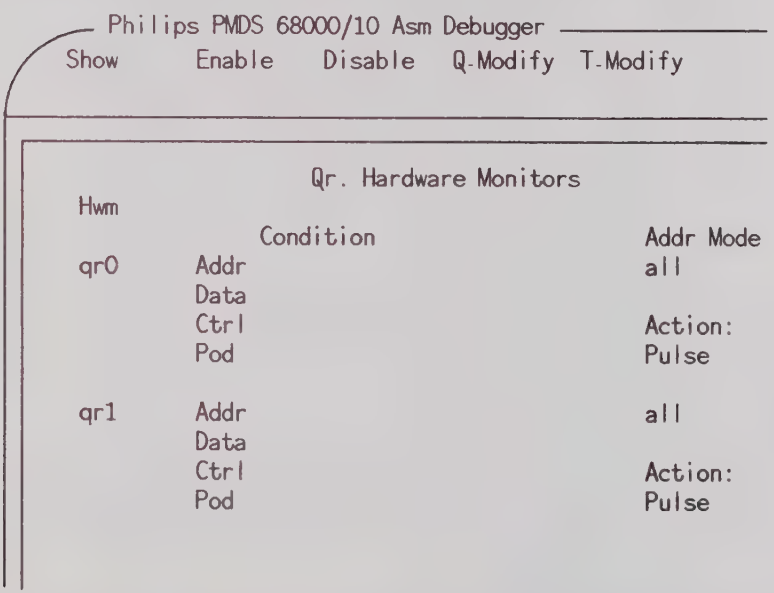
Philips PMDS 68000/10 Asm Debugger

Show Enable Disable Modify List

Line	Md	Addr	Data	Opcode	Operands	Control	Pod
-17	up	000078	3001	move.w	d1,d0	mr/pr	
-16	ud	001ffd	4c4c			mw	
-15	up	00007a	0c81	cmpi.l	#0x0,d1	mr/pr	
-14	up	00007c	0000			mr/pr	
-13	up	00007e	0000			mr/pr	
-12	up	000080	5bc9	dbmi	d1,0x88	mr/pr	
-11	up	000082	0004			mr/pr	
-10	up	000086	0838	btst	#0x1,0x1ffc	mr/pr	
-9	up	000088	0001			mr/pr	
-8	up	00008a	1ffc			mr/pr	
-7	up	00008c	67f8	beq.s	0x86	mr/pr	
-6	ud	001ffc	ff02			mr	
-5	up	00008e	11d9	move.b	(a1)+,0x1fff	mr/pr	
-4	up	000090	1fff			mr/pr	
-3	ud	010000	ffff			mr	
-2	up	000092	60e6			mr	
-1	ud	001fff	ffff			mw	
0	up	000094	4cdf			mr	

Line −3 shows us that the program tried to read from the guarded memory location 0x10000. As the error occurs when trying to load register *a1*, we set a condition so that the emulation halts when *a1* is loaded. From the assembly listing at the end of the tutorial it can be seen that this is done at address 0x56. Therefore, emulation should be halted immediately after this instruction has been executed. Thus the condition must be set for the address of the following instruction which is located at address 0x5a. A hardware monitor command can be used to set a condition on which program execution must be halted.

Therefore, we select the /dhq (*Debugger HwMonitor Q-Modify*) command which deals with the hardware monitor *qr0*.



qr0 is the name of the hardware monitor. Addr, Data, Pod and Ctrl can be used to specify a condition. We set Addr to 0x5a. As we want program execution to halt when address 0x5a has been reached, move the cursor to *pulse* and set Action to break using the cycle key +.

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Q-Modify  T-Modify

                    Qr. Hardware Monitors
Hwm
                    Condition                    Addr Mode
qr0       Addr 0x5a                              all
          Data
          Ctrl                                   Action:
          Pod                                    Break

qr1       Addr                                   all
          Data
          Ctrl                                   Action:
          Pod                                    Pulse
```

When pressing Enter twice, the command is executed and the following output is displayed:

```
Philips PMDS 68000/10 Asm Debugger
Show      Enable    Disable   Q-Modify  T-Modify

hwmon qr0 = all 0x00005a, "xxxxxxxxxxxxxxxx", "xxxx"
          break
hwmon qr0 : enabled
```

The strings containing only x's are bit strings that are set to don't care (this is what the series of x's indicate). As such, these

strings do not affect the condition.

Before rerunning the program, the processor has to be reset. This is done by entering /tp to select the *Target Processor* command, pressing cursor left twice to preselect *P-Reset* and finally pressing [Enter] to execute the command.

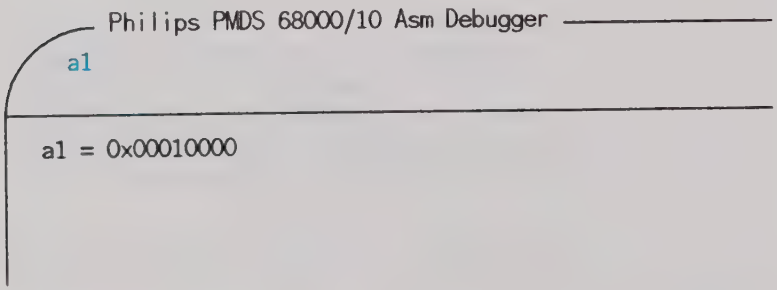
```
Philips PMDS 68000/10 Asm Debugger
Run        Halt       P-Reset    Wait

Target processor reset
```

The *Target Processor Run* command may now be entered by pressing /tp [Enter] . Note that the values of the run options are the same as the last time. New is thus set to *Yes*, and *Addr* is set to 0x30.

When the condition specified by the hardware monitor *qr0* is met, execution will halt and the following will be displayed:

```
Philips PMDS 68000/10 Asm Debugger
Run        Halt       P-Reset    Wait

Emulation started
Break : qr0
Emulation halted
```

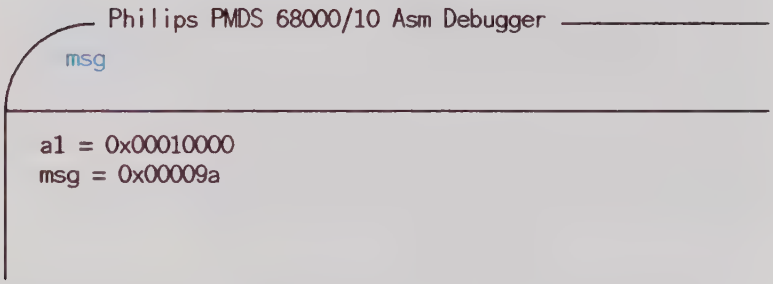
Direct input mode can be used to display the value of objects such as registers and symbols defined by the user in his program. Direct input mode is invoked when pressing the Dir Mode key. The menu window is replaced by an empty line on which an object may be entered. When pressing Enter , the value of the specified object is displayed in the default window. Consequently, the content of *a1* may be examined by pressing the Dir Mode key, typing *a1*, and pressing Enter :



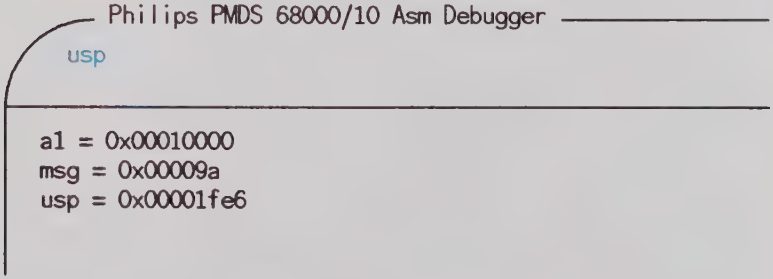
This shows that register *a1* has been loaded with address 0x10000. As *a1* should contain the address of the string to be output, we check the address of this string.

At the moment, the default display window is set to *overwrite* mode. By pressing the Append key, the mode is set to *append*. In this case, the output of the last entered command will be appended to the output of the previous commands.

After invoking the direct mode by pressing the Dir Mode key, the symbol *msg* may be entered to find the address of the string:



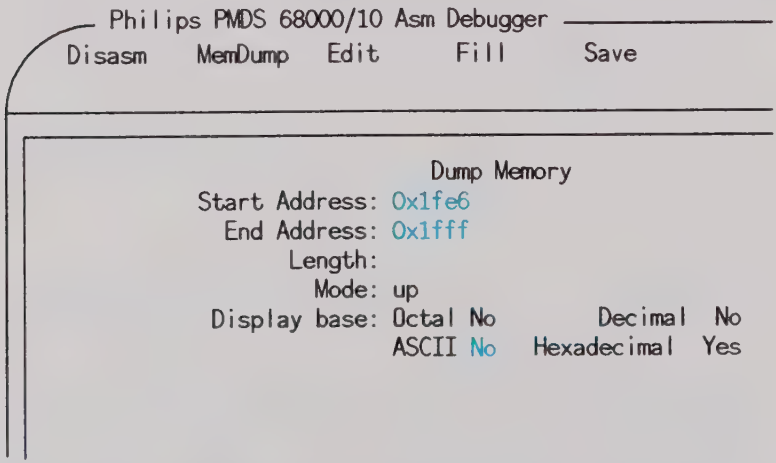
From this it can be seen that register *a1* has a different address than that of the string to be output. The contents of program memory is now examined to see if it has been corrupted. First of all, let us find out the current address of the user stack pointer *usp* by pressing Dir Mode , entering *usp*, and pressing Enter .



When pressing the Make Wnd key, a new display window is created by splitting the area of the current display window in two. The command output displayed in the top window does not change. Initially the bottom window is empty. When a new command is entered, its output will be displayed in the bottom window.

The contents of memory around this position can be examined by entering a /tmm (*Target Memory MemDump*) command, and

setting the options Start Address and End Address to 0x1fe6 and 0x1fff respectively. The ASCII option is set to *No* by means of the cycle key –.



When pressing Enter twice, the command is executed and the following output is displayed:

```
Philips PMDS 68000/10 Asm Debugger
 Disasm    MemDump   Edit      Fill      Save

a1 = 0x00010000
msg = 0x00009a
usp = 0x00001fe6


address     |                    hex
-------------------------------------------------------
up 0x001fe6 |                    00 00 00 00 00 00 00 48 00 00
up 0x001ff0 | 00 01 00 00 00 9a 00 00 00 0a 00 00 02 4c 00 ff
```

From this output we can see that the address of the string to be output was started at 0x1fe6 + 0x10, as was specified in line 56 of our program (move.l 10(sp), a1). Since the move.l statement gives four bytes from the offset, it can be seen that this does in fact give us address 0x10000 and not 0x9a (the address of the string to be output). Therefore, the offset specified in line 56 must be increased by 2:

move.l 12(sp),a1

5.10 Exiting from the Debugger and Patching

This is the end of the debug session, which can be terminated by entering:

/q
Enter

The log files that have been opened at the beginning of the debugging session are closed.

The files *sample.cmd* and *sample.dis* are present in the current directory and contain a complete record of all the debugging commands entered in our session, together with the response from the Debugger.

Remember that the source program should be corrected and re-assembled once the Debugger has terminated.

5.11 Program Listing

```
00000001                                    FD = 1
0000000A                                    NOCHARS = 10
                                            .sect vectors
                                            .base 0x0
00000000  0000                      r1:     .data2 0
00000002  1FFA                              .data2 stack
00000004  0000                      r2:     .data2 0
00000006  0030                              .data2 start

                                            .sect program
                                            .base 0x30

                                            .sect .text

                                            .sect .data
0000009A  5043503A203638303030      msg:    .ascii 'PCP: 68000'
000000A4  000A                      nochr:  .data2 NOCHARS

                                            .sect .bss
                                            .base 0x1000
                                            .space 0xffa
00001FFA                                    stack = .

                                            .sect program
                                            .define __write
00000030  3E7C1FFA                  start:  move #stack, a7
00000034  2F3800A4                          move.l nochr, -(sp)
00000038  2F3C0000009A                      move.l #msg, -(sp)
0000003E  2F3C00000001                      move.l #FD, -(sp)
00000044  4EB8004E                          jsr __write
00000048  4FEF000A                          lea 10(sp), sp
0000004C  60FE                      0:      bra 0b
```

```
00001FFC                        ACIA = 0x1ffc
00000003                        RESET = 0x03
0000004C                        MAGIC = 0x4C
                                .sect .text
0000004E 48E70002       _write: movem.l a6, -(sp)
00000052 222F0010               move.l 16(sp), d1
00000056 226F000A               move.l 10(sp), a1
0000005A 202F0008               move.l 8(sp), d0


0000005E 11FC00031FFD           move.b #RESET, ACIA+1
00000064 11FC00021FFC           move.b #2, ACIA
0000006A 303C03E8               move.w #1000, d0
0000006E 51C8FFFE       wait:   dbra d0, wait
00000072 11FC004C1FFD           move.b #MAGIC, ACIA+1

00000078 3001                   move d1, d0
0000007A 0C8100000000   0:      cmp.l #0, d1
00000080 5BC90004               dbmi d1, notend
00000084 600E                   bra 0f
                        notend:
00000086 083800011FFC   nready: btst #1, ACIA
0000008C 67F8                   beq nready
0000008E 11D91FFF               move.b (a1)+, ACIA+3
```

CONTENTS

		Page
6	HARDWARE REFERENCE GUIDE	6-3
6.1	Introduction	6-3
6.2	Physical Dimensions	6-3
6.3	Environmental Conditions	6-3
6.4	Electrical Data	6-5
6.5	DC Specifications For Emulation Probe	6-6
6.6	Timing Specifications For Emulation Probe	6-10
6.6.1	Clock in MDS (12.5MHz)	6-10
6.6.2	Clock in Prototype (10MHz)	6-12
6.7	Extended Control Lines	6-16

CHAPTER 6

HARDWARE REFERENCE GUIDE

6.1 Introduction

These specifications are valid after the instrument has warmed up for 10 minutes (reference temperature 22 degrees C). Properties expressed as a range are guaranteed by the manufacturer; other values are typical and represent the characteristics of the average instrument.

6.2 Physical Dimensions

Height	:	52mm
Width	:	296mm
Depth	:	256mm
Weight	:	2.2kg

6.3 Environmental Conditions

Temperature

Rated range of use	:	+10 deg C to +35 deg C

Limited range of use	:	+ 5 deg C to +45 deg C
Range for storage and transport	:	−40 deg C to +70 deg C

Humidity

Rated range of use	:	20% to 80% (no condensation)

Pressure

Rated range of use	:	up to 3500m
Range for storage and transport	:	up to 6500m

Vibration

Rated range of use	:	negligible value
Range for storage and transport	:	0.7mm/pp max 50m/s/s (freq range 10 to 55 Hz)

Mechanical Shock

Rated range of use	:	negligible value
Range for storage and transport	:	300m/s/s (max. 18 shocks with pulse width of 11 ms)

6.4 Electrical Data

Power supply

+ 5V at 2.8A
+12V at 90mA
−12V at 30mA

Power consumption

15W

6.5 DC Specifications For Emulation Probe

LOAD/DRIVE 68000-68010

PIN	INPUT LOAD		OUTPUT DRIVE		SIGNAL
	Iin h (uA)	Iin l (uA)	Iout h (mA)	Iout l (mA)	
1	20	-200	-15	24	D04
2	20	-200	-15	24	D03
3	20	-200	-15	24	D02
4	20	-200	-15	24	D01
5	20	-200	-15	24	D00
6	--	--	-15	24	ASN
7	--	--	-15	24	UDSN
8	--	--	-15	24	LDSN
9	--	--	-15	24	R/WN
10	20	-200	--	--	DTACKN
11	--	--	-15	24	BGN
12	20	-200	--	--	BGACKN

PIN	INPUT LOAD		OUTPUT DRIVE		SIGNAL
	Iin h (uA)	Iin l (uA)	Iout h (mA)	Iout l (mA)	
13	20	-200	--	--	BRN
14	--	--	--	--	Vcc
15	20	-200	--	--	CLK
16	--	--	--	--	GND
17	40	-1600	-0.4	1.6	HALTN
18	40	-1600	-0.4	5	RESETN
19	--	--	-15	24	VMAN
20	--	--	-15	24	E
21	20	-200	--	--	VPAN
22	20	-200	--	--	BERRN
23	20	-360	--	--	IPL2N
24	20	-360	--	--	IPL1N
25	20	-360	--	--	IPL0N
26	--	--	-15	24	FC2
27	--	--	-15	24	FC1
28	--	--	-15	24	FC0
29	--	--	-15	24	A01
30	--	--	-15	24	A02

PIN	INPUT LOAD		OUTPUT DRIVE		SIGNAL
	Iin h (uA)	Iin l (uA)	Iout h (mA)	Iout l (mA)	
31	--	--	-15	24	A03
32	--	--	-15	24	A04
33	--	--	-15	24	A05
34	--	--	-15	24	A06
35	--	--	-15	24	A07
36	--	--	-15	24	A08
37	--	--	-15	24	A09
38	--	--	-15	24	A10
39	--	--	-15	24	A11
40	--	--	-15	24	A12
41	--	--	-15	24	A13
42	--	--	-15	24	A14
43	--	--	-15	24	A15
44	--	--	-15	24	A16
45	--	--	-15	24	A17
46	--	--	-15	24	A18
47	--	--	-15	24	A19
48	--	--	-15	24	A20
49	--	--	--	--	Vcc
50	--	--	-15	24	A21

PIN	INPUT LOAD		OUTPUT DRIVE		SIGNAL
	Iin h (uA)	Iin l (uA)	Iout h (mA)	Iout l (mA)	
51	--	--	-15	24	A22
52	--	--	-15	24	A23
53	--	--	--	--	GN D
54	20	-200	-15	24	D15
55	20	-200	-15	24	D14
56	20	-200	-15	24	D13
57	20	-200	-15	24	D12
58	20	-200	-15	24	D11
59	20	-200	-15	24	D10
60	20	-200	-15	24	D09
61	20	-200	-15	24	D08
62	20	-200	-15	24	D07
63	20	-200	-15	24	D06
64	20	-200	-15	24	D05

6.6 Timing Specifications For Emulation Probe

6.6.1 Clock in MDS (12.5MHz)

		min	max	unit
1	clock period	80	250	ns
2	clock width low	35	125	ns
3	,, ,, high	35	125	ns
4	,, fall time	-	5	ns
5	,, rise time	-	5	ns
6	,, low to address	-	75	ns
6a	,, high to FC valid	-	80	ns
7	,, ,, to ad. data high imp.	-	80	ns
8	,, ,, to adr/FC invalid	20	-	ns
9	,, ,, to ASN low max	-	80	ns
9	,, ,, to DSN low max	-	130	ns
10	,, ,, to ASN low min	10	-	ns
10	,, ,, to DSN low min	50	-	ns
11	adr valid to ASN/DSN (read) low ASN low (write)	10	-	ns
11a	FC valid to ASN/DSN (read) low ASN low (write)	40	-	ns
12	clock low to ASN high	-	75	ns
12	,, ,, to DSN high	-	125	ns
13	ASN/DSN high to adr/FC invalid	10	-	ns
14	ASN/DSN width low (DSN read	160	-	ns
14a	DSN width low (write)	80	-	ns
15	ASN/DSN width high	65	-	ns
16	clock high to ASN high imp.	-	80	ns

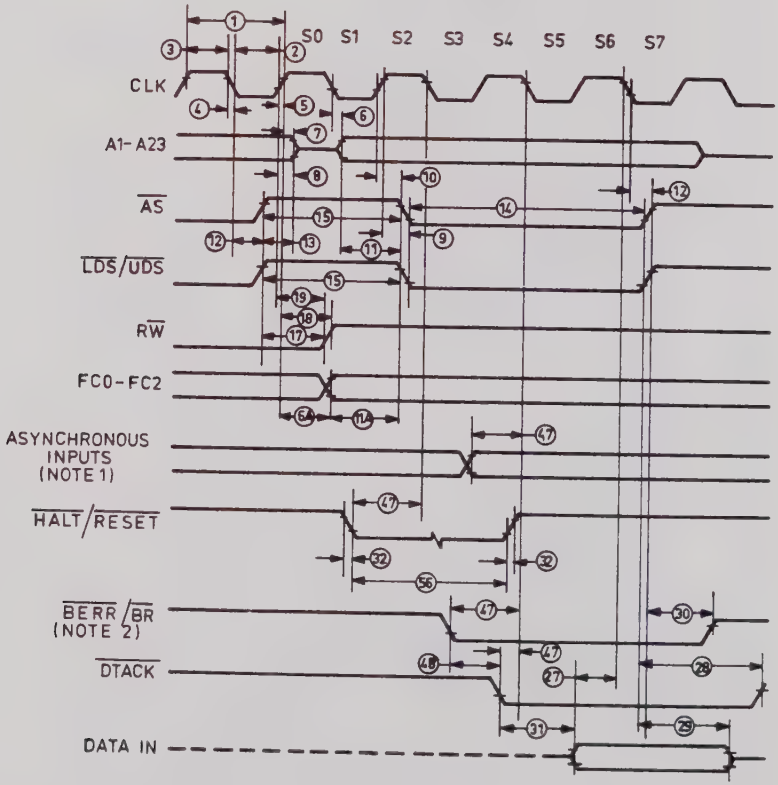
16	,, ,, to DSN high imp.	-	125	ns
17	ASN high to RWN high	10	-	ns
17	DSN high to RWN high	0	-	ns
18	clock high to RWN high (max)	-	85	ns
19	,, ,, to RWN high (min)	10	-	ns
20	,, ,, to RWN low	-	85	ns
21	adr valid to RWN low	10	-	ns
21a	FC valid to RWN low	45	-	ns
22	RWN low to DSN low (write)	30	-	ns
23	clock low to DATA out valid	-	70	ns
25	DSN high to DATA out invalid	0	-	ns
26	DATA out valid to DSN low	25	-	ns
27	DATA in to clock low (setup)	25	-	ns
28	ASN high to DATA high	-	110	ns
28	DSN high to DATA high	-	95	ns
29	DSN high to DATA invalid (hold)	0	-	ns
30	ASN/DSN high to BERR high	0	-	ns
31	DTACKN low to DATA in (setup)	-	90	ns
32	HALTN en RESETN input transition	0	200	ns
47	BGACKN, VPAN input setup time	40	-	ns
47	BRN ,, ,, ,,	55	-	ns
47	IPLO -2 ,, ,, ,,	50	-	ns
47	BERRN ,, ,, ,,	50	-	ns
47	DTACKN ,, ,, ,,	50	-	ns
48	BERRN low to DTACKN low (for 68000 only)	50	-	ns
48	DTACKN low to BERRN low (for 68010 only)	-	55	ns
53	DATA hold from clock high	0	-	ns
55	RWN to DATA bus high imp.	-	30	ns
56	HALTN/RESET pulse width (cp = clock periods)	10	-	cp

6.6.2 Clock in Prototype (10MHz)

		min	max	unit
1	clock period	80	250	ns
2	clock width low	35	125	ns
3	,, ,, high	35	125	ns
4	,, fall time	-	5	ns
5	,, rise time	-	5	ns
6	,, low to address	-	105	ns
6a	,, high to FC valid	-	110	ns
7	,, ,, to ad. data high imp.	-	110	ns
8	,, ,, to adr/FC invalid	30	-	ns
9	,, ,, to ASN low max	-	110	ns
9	,, ,, to DSN low max	-	160	ns
10	,, ,, to ASN low min	20	-	ns
10	,, ,, to DSN low min	60	-	ns
11	adr valid to ASN/DSN (read) low ASN low (write)	10	-	ns
11a	FC valid to ASN/DSN (read) low ASN low (write)	40	-	ns
12	clock low to ASN high	-	105	ns
12	,, ,, to DSN high	-	155	ns
13	ASN/DSN high to adr/FC invalid	10	-	ns
14	ASN/DSN width low (DSN read)	160	-	ns
14a	DSN width low (write)	80	-	ns
15	ASN/DSN width high	65	-	ns
16	clock high to ASN high imp.	-	110	ns
16	,, ,, to DSN high imp.	-	155	ns
17	ASN high to RWN high	10	-	ns
17	DSN high to RWN high	0	-	ns
18	clock high to RWN high (max)	-	115	ns
19	,, ,, to RWN high (min)	20	-	ns

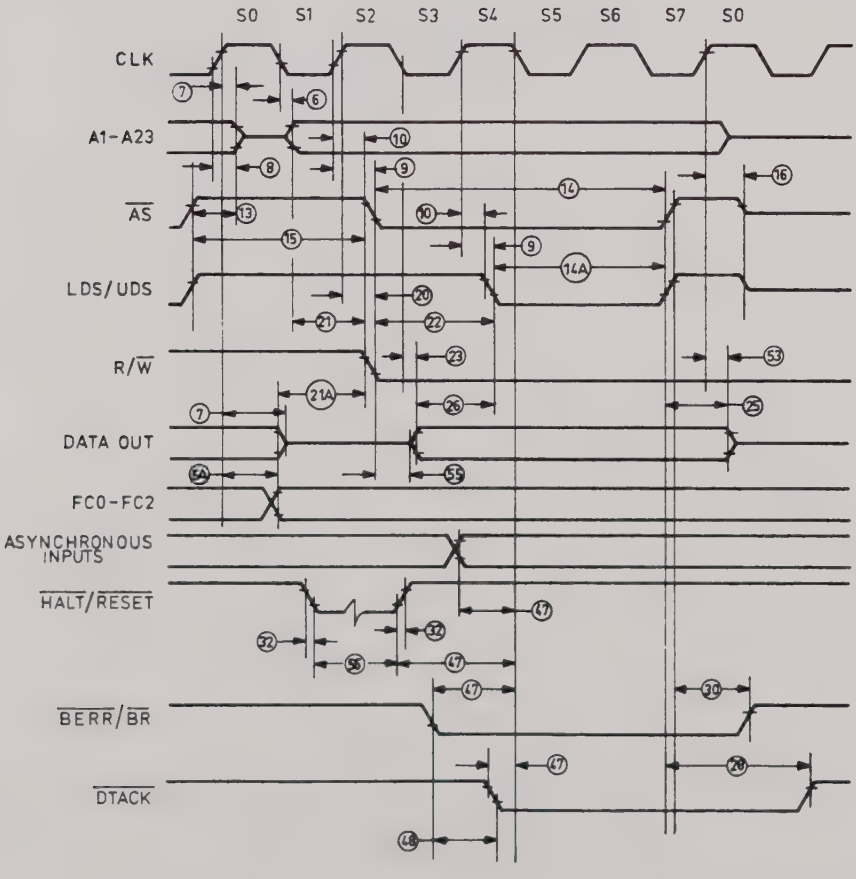
20	„ „ to RWN low	-	115	ns
21	adr valid to RWN low	10	-	ns
21a	FC valid to RWN low	45	-	ns
22	RWN low to DSN low (write)	30	-	ns
23	clock low to DATA out valid	-	100	ns
25	DSN high to DATA out invalid	0	-	ns
26	DATA out valid to DSN low	25	-	ns
27	DATA in to clock low (setup)	35	-	ns
28	ASN high to DATA high	-	110	ns
28	DSN high to DATA high	-	95	ns
29	DSN high to DATA invalid (hold)	0	-	ns
30	ASN/DSN high to BERR high	0	-	ns
31	DTACKN low to DATA in (setup)	-	90	ns
32	HALTN en RESETN input transition	0	200	ns
47	BGACKN, VPAN input setup time	40	-	ns
47	BRN „ „ „	55	-	ns
47	IPLO -2 „ „ „	50	-	ns
47	BERRN „ „ „	50	-	ns
47	DTACKN „ „ „	50	-	ns
48	BERRN low to DTACKN low (for 68000 only)	50	-	ns
48	DTACKN low to BERRN low (for 68010 only)	-	55	ns
53	DATA hold from clock high	0	-	ns
55	RWN to DATA bus high imp.	-	30	ns
56	HALTN/RESET pulse width (cp = clock periods)	10	-	cp

Read Cycle Timing For 68000



MA10548

Write Cycle Timing For 68000



6.7 Extended Control Lines

There are no Extended Control Lines for the 68000/68010.

APPENDIX A

Keyboard Layout for Standard PMDS-3 Keys on AT Console and TN-100 Keyboards

Keyboard Layout for Standard PMDS-3 Keys

on AT Console

Function Key Command	Function
Help	F1
Dir Mode	F2
Edit	F4
Annotate	Ctrl F5
MvBorder	Shift F5
Update	Alt F5
Append	F5
Run	F6
Del Wnd	Ctrl F7
Del Lvl	Ctrl F8
Make Wnd	Shift F7
Make Lvl	Shift F8
Next Wnd	Alt F7
Next Lvl	Alt F8
Cycle Wnd	F7
Cycle Lvl	F8
Halt	F9
Step	F10
Cmd Brk	Alt 8

on TN-100 Keyboards

Function Key Command	Function
Help	F1
Dir Mode	F2
Edit	F4
Annotate	F5
MvBorder	KP9
Update	KP6
Append	KP3
Run	F6
Del Wnd	F7
Del Lvl	F8
Split Wnd	KP7
Make Lvl	KP8
Next Wnd	KP4
Next Lvl	KP5
Cycle Wnd	KP1
Cycle Lvl	KP2
Halt	KP0
Step	Enter
Cmd Brk	KP–

Note: On TN-100 keyboards "KP" indicates that the key to use is to be found on the alternate key pad.

Other Keyboard Mappings Between AT Keyboard and TN-100-like Keyboards

PC AT Key	PM4499 Key
PgUp	Set Tab
PgDn	Clear Tab
CTRL-PgUp	Insert Line
CTRL-PgDn	Delete Line
Home	Delete Char
End	Insert Char

Note: Arrow keys are mapped to arrow keys.

Cycle keys remain + and − on all keyboards.

APPENDIX B

ASCII TABLE

ASCII Character	HEX	ASCII Character	HEX	ASCII Character	HEX	ASCII Character	HEX
NUL (↑@)	00	space	20	@	40	‘	60
SOH (↑A)	01	!	21	A	41	a	61
STX (↑B)	02	"	22	B	42	b	62
ETX (↑C)	03	*	23	C	43	c	63
EOT (↑D)	04	$	24	D	44	d	64
ENQ (↑E)	05	%	25	E	45	e	65
ACK (↑F)	06	&	26	F	46	f	66
BEL (↑G)	07	’	27	G	47	g	67
BS (↑H)	08	(	28	H	48	h	68
HT (↑I)	09	)	29	I	49	i	69
LF (↑J)	0A	.	2A	J	4A	j	6A
VT (↑K)	0B	+	2B	K	4B	k	6B
FF (↑L)	0C	’	2C	L	4C	l	6C
CR (↑M)	0D	-	2D	M	4D	m	6D
SO (↑N)	0E	.	2E	N	4E	n	6E
SI (↑O)	0F	/	2F	O	4F	o	6F
DLE (↑P)	10	0	30	P	50	p	70
DCI (↑Q)	11	1	31	Q	51	q	71
DC2 (↑R)	12	2	32	R	52	r	72
DC3 (↑S)	13	3	33	S	53	s	73
DC4 (↑T)	14	4	34	T	54	t	74
NAK (↑U)	15	5	35	U	55	u	75
SYN (↑V)	16	6	36	V	56	v	76
ETB (↑W)	17	7	37	W	57	w	77
CAN (↑X)	18	8	38	X	58	x	78
EM (↑Y)	19	9	39	Y	59	y	79
SUB (↑Z)	1A	:	3A	Z	5A	z	7A
ESC	1B	;	3B	[	5B	{	7B
FS	1C	<	3C	\	5C	|	7C
GS	1D	=	3D	]	5D	}	7D
RS	1E	>	3E	^ or ↑	5E	~	7E
US	1F	?	3F	_	5F	DEL	7F

Note: Control key equivalences are in parentheses.

INDEX

abort action command 4-7
abort command 1-13
action abort command 4-7
action commands 2-34
action end command 4-8
action if command 4-9
action list 1-10 2-35
action pause command 4-12
action return command 4-14
action while command 4-16
address mode 4-123
Adeb68010 1-3
analysis commands 2-22
annotate border command 4-19
ascii format 4-200
ASCII table B-2
assembly format 4-200
AT console layout A-2
binary format 4-200
block mapping 2-7
border annotate command 4-19
border move command 4-22
break 2-11,12 4-73,82
break key, command 1-17
buserror 2-10 4-76
/ccm 4-25
/ccs 4-28
centre-tracing 2-17
change directory 2-31 4-179
character constants 3-8
chdir command 4-179
/cid 4-87
/cie 4-88
/cis 4-89
class 4-123
clear progsymb command 4-149
clock 4-191
clock command 2-4
clock, microprocessor development system 6-10
clock modify command 4-25

clock, prototype 6-12
clock show command 4-28
clockfail 2-10 4-76
/cmd 4-117
Cmd Brk key 1-17
/cme 4-118
/cmi 4-119
/cmm 4-122
/cms 4-127
command all command 4-29
command break key 1-17
command command command 4-32
command file comment lines 2-33
command files 2-30
command history 3-6
comment line 3-4
comment lines, command file 2-33
configuration group 2-4
console keys A-2
count disable command 4-35
count enable command 4-36
count modify command 4-37
count show command 4-42
/cpm 4-137
/cps 4-139
current window 1-7
dc specifications for emulation probe 6-6
/dcd 4-35
/dce 4-36
/dcm 4-37
/dcs 4-42
debugger group 2-10
debugging facilities 1-3
debugging, symbolic 1-26
.debugrc 1-4
default window 1-7
delete file 2-31
/dhd 4-63
/dhe 4-66
/dhq 4-69
/dhs 4-76
/dht 4-79
direct command format 1-10
direct input mode 3-3
direct mode commands 3-3
directory, change 2-31 4-179
disasm memory command 4-101

INDEX

abort action command 4-7
abort command 1-13
action abort command 4-7
action commands 2-34
action end command 4-8
action if command 4-9
action list 1-10 2-35
action pause command 4-12
action return command 4-14
action while command 4-16
address mode 4-123
Adeb68010 1-3
analysis commands 2-22
annotate border command 4-19
ascii format 4-200
ASCII table B-2
assembly format 4-200
AT console layout A-2
binary format 4-200
block mapping 2-7
border annotate command 4-19
border move command 4-22
break 2-11,12 4-73,82
break key, command 1-17
buserror 2-10 4-76
/ccm 4-25
/ccs 4-28
centre-tracing 2-17
change directory 2-31 4-179
character constants 3-8
chdir command 4-179
/cid 4-87
/cie 4-88
/cis 4-89
class 4-123
clear progsymb command 4-149
clock 4-191
clock command 2-4
clock, microprocessor development system 6-10
clock modify command 4-25

clock, prototype 6-12
clock show command 4-28
clockfail 2-10 4-76
/cmd 4-117
Cmd Brk key 1-17
/cme 4-118
/cmi 4-119
/cmm 4-122
/cms 4-127
command all command 4-29
command break key 1-17
command command command 4-32
command file comment lines 2-33
command files 2-30
command history 3-6
comment line 3-4
comment lines, command file 2-33
configuration group 2-4
console keys A-2
count disable command 4-35
count enable command 4-36
count modify command 4-37
count show command 4-42
/cpm 4-137
/cps 4-139
current window 1-7
dc specifications for emulation probe 6-6
/dcd 4-35
/dce 4-36
/dcm 4-37
/dcs 4-42
debugger group 2-10
debugging facilities 1-3
debugging, symbolic 1-26
.debugrc 1-4
default window 1-7
delete file 2-31
/dhd 4-63
/dhe 4-66
/dhq 4-69
/dhs 4-76
/dht 4-79
direct command format 1-10
direct input mode 3-3
direct mode commands 3-3
directory, change 2-31 4-179
disasm memory command 4-101

display windows 1-9
displaying, file 2-28
dma 2-10 4-77
/dsd 4-167
/dse 4-169
/dsm 4-171
/dsr 4-175
/dss 4-177
/dtd 4-187
/dte 4-188
/dtl 4-189
/dtm 4-191
/dts 4-195
dump all command 4-45
dump load command 4-49
dump memory command 4-111
dumping, file 2-29
dyadic operators 3-9
edit all command 4-53
edit command command 4-57
edit source command 4-60
editing, file 2-27
editor parameter 1-5
end action command 4-8
error handling 1-12
error window 1-9
event 4-38
event counter (rtec), real-time 2-18
expression evaluation 3-10
expressions 3-7
extended control lines 6-16
/fca 4-29
/fcc 4-32
/fda 4-45
/fdl 4-49
/fea 4-53
/fec 4-57
/fes 4-60
file, delete 2-31
file displaying 2-28
file dumping 2-29
file editing 2-27
file loading 2-28
file, remove 2-31
file specification (suffix) 2-26
file suffix 2-26
file viewing 2-28

files group 2-26
fill memory command 4-107
/fla 4-95
/fll 4-98
/fpa 4-131
/fpl 4-134
from 4-39
/fsc 4-179
/fsp 4-181
/fsr 4-183
function keys 1-24 A-2
/fva 4-203
/fvc 4-206
/fvs 4-208
halt processor command 4-141
haltline 2-10 4-77
hardware monitors 2-10
help facility 1-6
help window 1-9
history buffer 3-3
humidity 6-4
hwmonitor command 2-10
hwmonitor disable command 4-63
hwmonitor enable command 4-66
hwmonitor Q-modify command 4-69
hwmonitor show command 4-76
hwmonitor T-modify command 4-79
if command 2-35 4-9
input source stack 1-11
input sources 1-10,11
instr 2-13
interrupts command 2-4
interrupts disable command 4-87
interrupts enable command 4-88
interrupts show command 4-89
invert memorymap command 4-119
invoking the debugger 1-3
keyboard input 1-15
keyboard layout A-2
length 4-123
level commands 2-33
level cycle command 4-91
level delete command 4-92
level make command 4-93
load all command 4-95
load load command 4-98
loading, file 2-28

log file 3-3
logging 2-32
mapping, block 2-7
match point 2-11
maximum length, object name 3-8
maximum number of levels 4-93
maximum number of windows 4-213
mechanical shock 6-5
memdump memory command 4-111
memory commands 2-22
memory disasm command 4-101
memory edit command 4-104
memory fill command 4-107
memory memdump command 4-111
memory, real-time trace 2-16
memory save command 4-114
memory type 4-120
memory types 2-4
memorymap command 2-4
memorymap disable command 4-117
memorymap enable command 4-118
memorymap invert command 4-119
memorymap modify command 4-122
memorymap show command 4-127
menu based commands 2-3
menu mode 3-3
menu window 1-8,15
monadic operators 3-9
monitors 2-10
monitors, hardware 2-10
monitors, software 2-13
move border command 4-22
notready 2-10 4-77
number representations 3-8
object 1-26
object name maximum length 3-8
objects group 2-24
octal format 4-200
/opc 4-149
operands 3-7
operators 3-8
option windows 1-9,18,19
/ovd 4-197
/ove 4-199
patch all command 4-131
patch load command 4-134
path 4-181

path command 4-181
pause action command 4-12
physical dimensions 6-3
pod command 2-4 4-137
pod show command 4-139
pod0 4-137,139
pop-up windows 1-9
post-tracing 2-17
power consumption 6-5
power supply 6-5
powerfail 2-10 4-77
pre-mapping 2-7
p-reset command 2-22
P-reset processor command 4-142
pressure 6-4
pre-tracing 2-17
processor halt command 4-141
processor P-reset command 4-142
processor run command 4-144
processor wait command 4-147
program objects 1-26 2-24
program symbols command 2-25
progsymb clear command 4-149
protection 4-124
pulse 2-12 4-73,83
/q 4-151
qr0 2-10,13 4-76
qr1 2-10,13 4-76
qual 4-192
quit command 4-151
radix 4-200
real-time event counter 2-13
real-time event counter (rtec) 2-18
real-time trace memory 2-16
recoverable errors 1-12
registers modify command 4-153
registers show command 4-156
remove file 2-31
rerun 2-10 4-77
reset processor command 4-142
reset target processor 2-22
resetline 2-10 4-77
resume session command 4-164
return action command 4-14
return command 1-13
rtec 2-13,16,18
run processor command 4-144

running a program 2-21
/saa 4-7
/sae 4-8
/sai 4-9
/sap 4-12
/sar 4-14
save memory command 4-114
/saw 4-16
/sba 4-19
/sbm 4-22
segment 2-7 4-124
services group 2-32
session command 2-32
session display command 4-159
session modify command 4-161
session resume command 4-164
session suspend command 4-165
shell escape command 3-4
/slc 4-91
/sld 4-92
/slm 4-93
software monitors 2-13
source stack, input 1-11
sources, input 1-11
/ssd 4-159
/ssm 4-161
/ssr 4-164
/sss 4-165
start address 4-123
start-address 4-120
status box, window 2-34
step size 4-145
suffix, file 2-26
suspend session command 4-165
/swc 4-211
/swm 4-213
swmonitor command 2-13
swmonitor disable command 4-167
swmonitor enable command 4-169
swmonitor modify command 4-171
swmonitor remove command 4-175
swmonitor show command 4-177
/sws 4-215
symbolic debugging 1-26
system chdir command 4-179
system commands 2-31
system path command 4-181

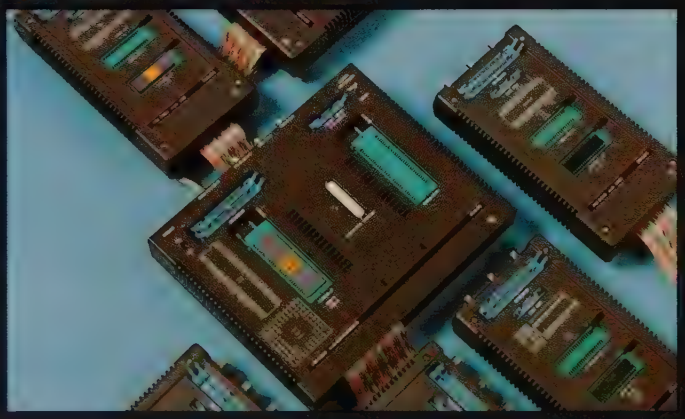
system remove command 4-183
target group 2-21
target processor, reset 2-22
temperature 6-3
/tmd 4-101
/tme 4-104
/tmf 4-107
/tmm 4-111
/tms 4-114
TN-100 layout A-2
to 4-39
/tph 4-141
/tpp 4-142
/tpr 4-144
/tpw 4-147
tr0 2-10,13 4-76
tr1 2-10,13 4-76
trace back command 2-24
trace disable command 4-187
trace enable command 4-188
trace facility 2-16
trace list command 4-189
trace modify command 4-191
trace show command 4-195
triggers 2-13
/trm 4-153
/trs 4-156
Unix commands 3-4
unrecoverable errors 1-13
value 4-38
variables command 2-24
variables describe command 4-197
variables examine command 4-199
vibration 6-4
view all command 4-203
view command command 4-206
view parameter 1-5
view source command 4-208
viewing, file 2-28
wait command 2-21
wait processor command 4-147
while command 2-35 4-16
wild card pattern 2-26
window 4-193
window cycle command 4-211
window delete command 4-212
window make command 4-213

window managing commands 2-33
window state command 4-215
window status box 2-34
windows 1-7
Xenix commands 3-4

Test & Measurement

PHILIPS

ASSEMBLY DEBUGGER 68000/68010 HARDWARE AND SOFTWARE INSTALLATION



PMDS

A publication of

Philips I&E

T & M - DTS Documentation Department,

Eindhoven, The Netherlands

Publication number 4822 871 90792

Copyright © 1988 Philips Export BV

All rights reserved. Reproduction in whole or part is prohibited without the written consent of the copyright owner.

UNIX is a trademark of Bell Laboratories

Printed in The Netherlands.

MANUAL PURPOSE AND STRUCTURE

PURPOSE

This manual is aimed at the user who has to install the Philips Microcomputer Development System 68010/68000 Assembly Debugger.

The manual must be used in conjunction with the *System Manager's Manual* and *Debugger Manual* for the system (PEDS or PMDS-II).

STRUCTURE

This manual contains information for the installation of the 68010/68000 Debugger. It consists of one chapter. The first part of the chapter gives details of the hardware components and how to install them. The second part describes the software installation procedure.

CONTENTS

		Page
1	INTRODUCTION AND INSTALLATION	1-3
1.1	Overview	1-3
1.2	Supported Microprocessors	1-3
1.3	Hardware Installation	1-5
1.3.1	MAB Emulation Memory	1-6
1.3.2	Installing Fast Emulation Memory Extension	1-6
1.3.3	The No Break Switch	1-7
1.3.4	The Microprocessor Adaptor Box (MAB)	1-9
1.3.5	Installing the MAB	1-11
1.3.6	The Test Pod	1-14
1.3.7	Break Trace Expansion Cable	1-15
1.4	Software Installation	1-17

INDEX

CONVENTIONS USED IN THIS MANUAL

<CR>	Indicates the *ENTER* key of the keyboard.
CTRL-x	Indicates that the *CTRL* key of the keyboard and the following key, in this case x, are to be pressed simultaneously.
make	Indicates a command. All commands are shown in **bold**.
	Italic characters are used throughout the manual for references to other manuals, filenames and commands used in text and markings on the hardware.

CONTENTS

		Page
1	INTRODUCTION AND INSTALLATION	1-3
1.1	Overview	1-3
1.2	Supported Microprocessors	1-4
1.3	Hardware Installation	1-4
1.3.1	MAB Emulation Memory	1-5
1.3.2	Installing Fast Emulation Memory Extension	1-6
1.3.3	The No Break Switch	1-7
1.3.4	The Microprocessor Adaptor Box (MAB)	1-9
1.3.5	Installing the MAB	1-11
1.3.6	The Test Pod	1-14
1.3.7	Break Trace Expansion Cable	1-15
1.4	Software Installation	1-17

CHAPTER 1

INTRODUCTION AND INSTALLATION

1.1 Overview

The 68000/68010 Assembly Screen Oriented Debugger (known simply as 'the Debugger') is an interactive software development tool that must be used with the 68000/68010 Microprocessor Adaptor Box (MAB). The Debugger provides the software control for the testing of partially or totally completed prototype hardware and software modules for 68000 and 68010 microprocessors.

This chapter describes the installation procedure for the 68000/68010 microprocessor Assembly Debugger. Installation of debugger hardware is given in the *System Manager's Manual* for PMDS-II or PEDS (called PMDS here) as appropriate. This chapter also describes specific points in the installation procedure for the 68000/68010 microprocessor Assembly Debugger hardware where they differ from the standard.

1.2 Supported Microprocessors

The following microprocessors are supported by this Debugger:

- MC68000
- MC68010

1.3 Hardware Installation

The following sections provide additional descriptive or installation information about each hardware component.

The system should first be installed as described in the *PMDS System Manager's Manual.* Ensure that a CMC board is installed. The Microprocessor Adaptor Box (MAB) and associated components can then be installed.

The following summary shows the hardware components of the debuggers and a reference to the installation instructions:

- **68000/68010 Microprocessor Adaptor Box (MAB).**
  Installation in *PMDS System Manager's Manual.*
- **The No Break Switch** (optional)
  Installation below. Skip the section if not installing the No Break Switch.
- **The Test Pod** (alternative)
  Installation in *PMDS Debugger Manual.* See also comments below.

- **The Break Trace Expansion Cable** (alternative)
  Installation below. Skip the section if not installing this cable.
- **The Fast Emulation Memory Extension Unit** (optional)
  Installation below. Skip the section if not installing this unit.

When installing the PMDS system, please take into account the lengths of cables supplied and ventilation requirements as detailed in the *PMDS System Manager's Manual.* Not mentioned there is that the break-trace expansion cable connecting the MAB and the CMC-board is 200cm long.

1.3.1 MAB Emulation Memory

The standard system can use three types of memory:

- internal emulation memory located in the system unit. This memory is available in dynamic or static form.
- the memory on the target system.
- external emulation memory located in the MAB.

The last mentioned is static memory located in the optional Fast Emulation Memory Extension Unit. This can be installed on the MAB (as described below) and has a size of 32 Kbytes This enables emulation up to the maximum speed of the microprocessor, with an access time of approximately 70ns.

These memories are described fully in the *PMDS Debugger Manual.*

1.3.2 Installing Fast Emulation Memory Extension

The procedure to install a Fast Emulation Memory Extension Unit is as follows:

STEP 1

If the MAB has been installed previously, ensure that the power switch on the MAB is turned to the OFF position.

STEP 2

Line up the 50-way D-type plug on the unit with the 50-way D-type socket on the MAB and the four feet of the unit with the four rectangular holes in the MAB casing and push the unit gently onto the MAB.

STEP 3

Drop the retaining screw through the hole in the centre of the unit and tighten it. After a few turns the screw will drop into a second thread. Continue to tighten the screw until secure.

The Fast Emulation Memory Extension Unit is now installed on the MAB.

1.3.3 The No Break Switch

The No Break Switch is a small unit designed to fit between the MAB and the system unit. It is designed to prevent the incorrect use of the MAB for your system. The No Break Switch is only necessary when using another MAB that is not supplied with a power switch.

The procedure to install the No Break Switch is as follows:

STEP 1

Undo the two allen screws on the top cover of the unit and remove the cover.

STEP 2

Pull back the ejector 'ears' and insert the 50-way twin flat ribbon cable into the connectors as shown in Figure 1.1.

CAUTION

It is possible to attach these connectors incorrectly with catastrophic results. Ensure that the cables are attached as shown in Figure 1.1.

STEP 3

Replace the cover of the unit and insert the two allen screws.

The No Break Switch is now installed.

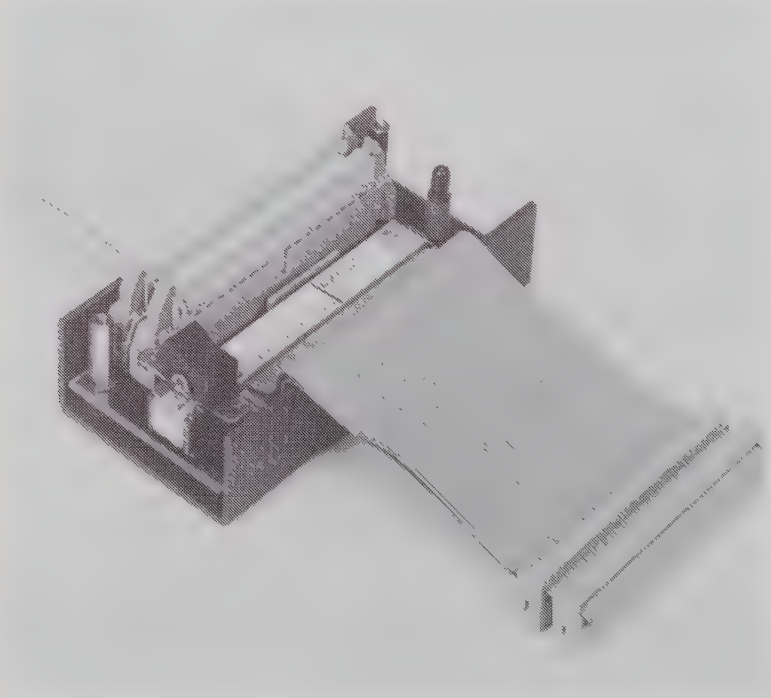


Figure 1.1 Connecting the Universal Debugger Cable to the No Break Switch.

1.3.4 The Microprocessor Adaptor Box (MAB)

The Microprocessor Adaptor Box (MAB) is the interface between the target microprocessor and the system unit. It can be connected to the target hardware for emulation or used in a stand alone simulation mode. The target microprocessor must be placed in the socket marked *TARGET μP* on the MAB and the *SELECT PROCESSOR* switch on the MAB set to the microprocessor used (68000 or 68010).

The 68000 and 68010 processors are available in two packings: 64-pin Dual In Line (DIL) and 10x10 pin-grid. The 'target processor' must be a DIL version; the Emulation Probe can have an adaptor fitted to allow for insertion into prototype circuits with either chip packaging.

Prototype hardware is debugged in emulation mode. Emulation is achieved by connecting the Emulation Probe to the microprocessor socket of the prototype hardware, if necessary through a DIL to pin-grid adaptor.

Programs can be tested without any application hardware in simulation mode, in which case the Debugger acts as a software development station. To work in simulation mode, the emulation probe must be connected to the simulation socket on the MAB.

The Emulation Probe is connected to either of the sockets by means of a flat ribbon cable that is 30cm long and has a 64-pin DIL plug at its end.

A BNC type connector (marked *TTL CLOCK IN*) is provided for the connection of an external clock source (external to the MAB and target).
This clock source can be selected by the debugger software. The clock frequency that is actually used by the MAB is the source frequency divided by 2. This frequency is limited by by the type

of memory that is used.

Two other BNC connectors (marked *TRIG 1 OUT* and *TRIG 0 OUT*) are provided for the output of trigger signals to external equipment.

The MAB also contains a 20-way connector marked *BREAK TRACE EXPANSION* which can be used to allow tracing of all 23 address lines and the *byte* line which indicates whether the CPU is reading from or writing to the lower or upper data bus.

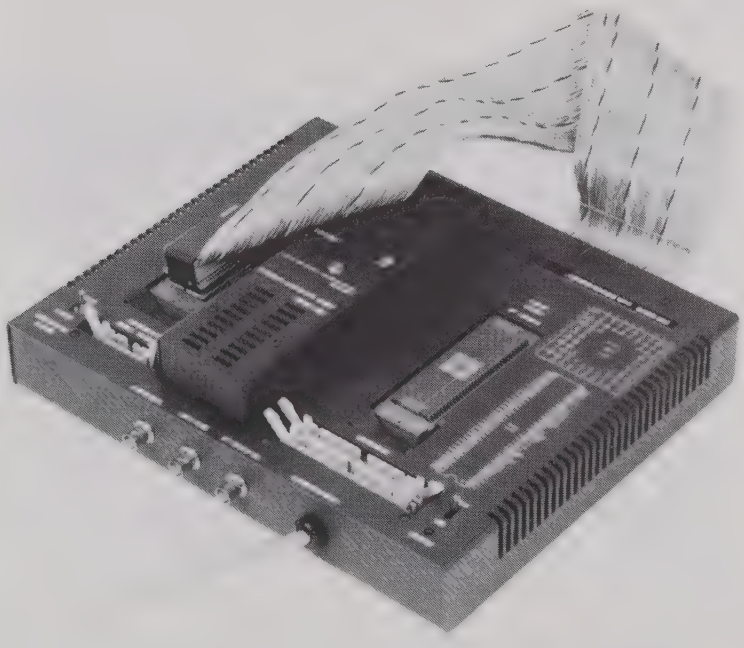


Figure 1.2 Microprocessor Adaptor Box (MAB)

1.3.5 Installing the MAB

The procedure to install a MAB is as follows:

STEP 1

The MAB is connected to the system unit using a 2 metre twin flat ribbon cable. Plug one end of this cable into the two 50-way sockets on the Real Time Board of the Universal Debugger as shown in the *PMDS Debugger Manual*. Each connector has a filled pin to ensure the correct polarization.

CAUTION

Excessive force used in attaching these connectors can break the filled pins, or bend the pins on the Real Time Board, resulting in incorrect connection which can have catastrophic results.

STEP 2

Ensure that the ON/OFF switch on the MAB is in the OFF position and that the *power* LED is unlit. If the No Break Switch is being used it can now be connected to the MAB.
If the No Break Switch is not being used the two remaining connectors of the UDC can now be connected.

CAUTION

It is possible to attach the connectors incorrectly onto the MAB with catastrophic results. Ensure that the cables are attached as shown in Figure 1.3.

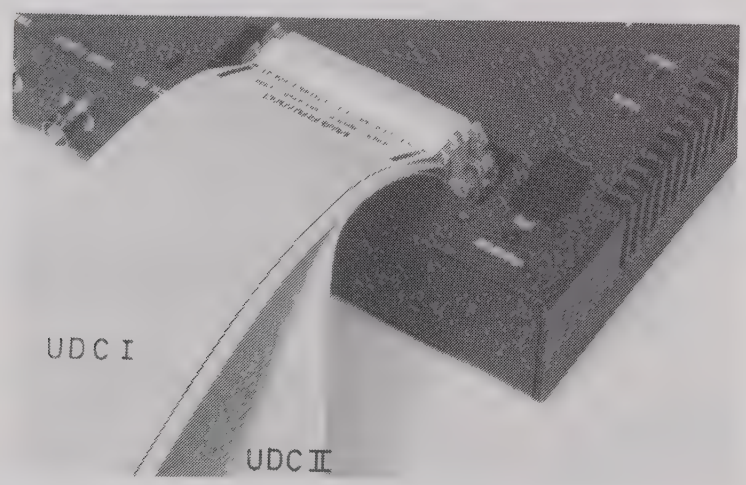


Figure 1.3 Connecting the UDC to the MAB

STEP 3

For simulation mode connect the Emulation Probe into the ZIF (Zero Insertion Force) socket marked *SIMULATION* on the MAB, ensuring that pin 1 of the Emulation Probe aligns with pin 1 (marked by a white dot) of the socket.

STEP 4

For emulation mode, connect the Emulation Probe to the microprocessor socket on the target system, ensuring that pin 1 on the probe corresponds to pin 1 on the socket. If the 'target processor' is a pin-grid version, a DIL to pin-grid adaptor must be used.

STEP 5

Connect the MAB power cable to the socket on the system unit marked *MAB POWER OUT*. (See *PMDS Debugger Manual* for details.)

STEP 6

Switch the power switch on the MAB to the ON position.

STEP 7

If connected, ensure the No Break Switch ON/OFF switch is in the ON position.

The MAB is now installed and when the software is installed, a debugging session can be started.

1.3.6 The Test Pod

The Test Pod is used to detect up to 8 signals in the target circuit in addition to those provided by the Emulation Probe. It

consists of 8 probes which may be connected to test hooks, IC clips, or the legs of a wire-wrap IC socket. The signals can be used to control the emulation, via the hardware monitors tr0, tr1, qr0 and qr1 (as described in Chapter 2 of the *Assembly Debugger 68000/68010 Manual*) and the event counter, or captured as history data by the trace memory.

The system can support one Test Pod or one Break Trace Expansion Cable (described below), connected to the system unit.

Further information and instructions for connecting the Test Pod are in the *PMDS Debugger Manual.*

1.3.7 Break Trace Expansion Cable

The Break Trace Expansion Cable is a separate flat cable which is provided to make a connection between the MAB and the system unit in place of the Test Pod connection.

This cable makes it possible to trace all 23 address lines and the *byte* line which indicates whether the CPU is reading from or writing to the lower or upper data bus. The possibilities for monitoring bus conditions and use of the *HwMonitor* commands become changed slightly when the Break Trace cable is connected. These changes are described within the *HwMonitor* commands in Chapters 2 and 4 of the *Assembly Debugger 68000/68010 Manual.*

Installing the Break Trace Expansion Cable

The procedure to install the Break Trace Expansion Cable is as follows:

STEP 1

The Break Trace Expansion Cable is terminated by a 20 way connector at each end. Fit one end into the Break Trace Expansion male socket on the MAB.

STEP 2

Fit the other end of the Break Trace Expansion Cable into the Test Pod connector on the CMC board (see *PMDS System Manager's Manual* for the location of this connector). The sockets and plugs have orientation lugs to prevent them being connected incorrectly.

1.4 Software Installation

The software is supplied on several floppy discs, PM4619M. It is advisable to load the new software when logged in as super-user in single user mode, to prevent *fsck(1)* running on a volatile file system.

The procedure to load the software[1] is:

STEP 1

Put the system in single user mode.

STEP 2

Log in as super-user.

STEP 3

Insert PM4619M floppy 1 into the disc drive.

1) Text for user to enter is shown in **bold**

STEP 4

Mount the floppy disc and copy the file *makesoi* with the commands:

```
/etc/mount /dev/dsk0.48 /mnt -r
cd /
cp /mnt/makesoi .
```

STEP 5

Unmount the floppy disc with the command:

```
/etc/umount /dev/dsk0.48
```

STEP 6

Execute the file makesoi:

```
/makesoi
```

STEP 7

Follow the instructions provided by this file, as shown on the screen. If anything goes wrong, try starting again at Step 6, makesoi.

MANUAL PURPOSE AND STRUCTURE

PURPOSE

This manual is aimed at the user who has to install the Philips Microcomputer Development System 68010/68000 Assembly Debugger.

The manual must be used in conjunction with the *System Manager's Manual* and *Debugger Manual* for the system (PEDS or PMDS-II).

STRUCTURE

This manual contains information for the installation of the 68010/68000 Debugger. It consists of one chapter. The first part of the chapter gives details of the hardware components and how to install them. The second part describes the software installation procedure.

STEP 8

When the process is finished, run the file system integrity check program fsck(1):

fsck

STEP 9

Reenter multi-user mode.

INDEX

adaptor box installation 1-9
break trace expansion cable installation 1-15
emulation memory installation 1-6
emulation mode connection 1-14
fast emulation memory extension installation 1-6
FEME installation 1-6
installation, break trace expansion cable 1-15
installation, emulation memory 1-6
installation, fast emulation memory extension 1-6
installation, hardware 1-5
installation, MAB 1-9
installation, microprocessor adaptor box 1-9
installation, no break switch 1-7
installation, software 1-17
installation, test pod 1-14
MAB installation 1-9
memory types 1-6
microprocessor adaptor box installation 1-9
no break switch installation 1-7
simulation mode connection 1-13
software installation 1-17
test pod installation 1-14
TRIG IN 1-10
TRIG OUT 1-10
TTL CLOCK IN 1-10

PHILIPS Sales and Service World-wide

Australia:
Philips Scientific & Industrial
Equipment Div. Centrecourt, 25–27 Paul
Street, North Ryde, Sydney N.S.W. 2113;
tel. 88 88 222

Austria:
Oesterreichische Philips Industrie
GmbH, Abteilung Industrie
Elektronik, Parttartgasse 32, A–1230, Wien;
tel. (0222) 841611

Belgium:
N.V. Philips & MBLE Associated S.A.
Scientific & Industrial Equipment Division
80 Rue des Deux Gares, Bruxelles;
tel. 2–5230000

Canada:
Philips Electronics Ltd.
601 Milner Avenue
Scarborough, Ontario M1B1Mb;
tel. (416) 292–5161.

Denmark:
Philips Elektronik Systemer A/S
Afd. Industri & Forskning
Strandlodsvej 4DK–2300–København S;
tel. (1) 572222; telex 31245.

Finland:
OY Philips AB, Kaivokatu 8
P.O. Box 255, 00101 Helsinki 10;
tel. 01–7271.

France:
S.A. Philips Industrielle et Commerciale
Science et Industry
105 Rue de Paris, 93002 Bobigny;
tel. 1–8301111

Germany (F.R.G.):
Philips GmbH
Unternehmensbereich Elektronik fuer
Wissenschaft und Industrie
3500, 35 Kassel; tel. 561–5010.

Great Britain:
Pye Unicam Ltd.
York Street
Cambridge CB1 2 PX;
tel. (0223) 358866.

Hong Kong:
Philips Hong Kong Ltd.
P.O.B. 2108, 27–29th floor Hopewell Centre
17 Kennedy Road, Hong Kong;
tel. 5–283298.

India:
Peico Electronics & Electricals Ltd.
PLOT 80, Bhosari Industrial Estate
Pune 411026;
tel. 82516/83547.

Ireland:
Philips Electrical (Ireland) Ltd.
Newstead, Clonskeagh
Dublin 14;
tel. 693355.

Italy:
Philips S.P.A., Sezione I&E T&M dept.
Viale Elvezia 2
20052 Monza;
tel. (039) 3635.1, telex 333343.

Netherlands:
Philips Nederland B.V.
Boschdijk 525, Gebouw VB
5600 MD, Eindhoven;
tel. (040) 782808.

New Zealand:
Philips Electrical Ind. Ltd.
Scientific & Industrial Division
68–86 Jervois Quay, P.O.Box 2097
Wellington C1; tel. 735735.

Norway:
Norsk Aktieselskap Philips
Dept. Industry and Telecommunication
Sandstuveien 70, P.O.Box 1, Mangelrud
Oslo 6 tel: 2–680200

PHILIPS Sales and Service World-wide

Portugal:
Philips Portuguesa S.A.R.L.
Av. Eng. Duarte Pacheco, Appartado 1331
Lisbon–1:
tel. (019) 683121.

Singapore:
Philips Singapore Private Ltd.
Professional Division, Lorong 1, Toa Payoh
P.O.Box 340, Toa Payoh Central Post Office
Singapore–1231:
tel. 2538811.

Spain:
Philips Iberica S.A.E.
Dpto Apparatos de Medida
Martinez Villergas 2
Madrid:
tel. 404–2200.

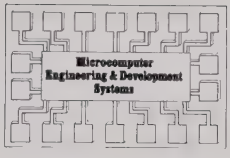
Sweden:
Svenska A.B. Philips
Div. Industrielektronik Avd. Matinstrument
11564 Stockholm:
tel. (08) 7821000.

Switzerland:
Philips AG, I&E Equipment
Allmendstrasse 140, Postfach 670
CH–8027 Zurich:
tel. (01) 488211.

Taiwan:
Philips Taiwan Ltd.
6th Floor, 150 Tun Hua N. Road
P.O.Box 22978, Taipei:
tel. (02) 712–0500.

U.S.A.:
Philips Test and Measurement Department Inc.
85, McKee Drive, Mahwah,
New Jersey 07430:
tel. 201–529–3800.

For countries not listed above:
Philips Export BV.
I&E Division Building HBS
P.O.Box 218,
5600 MD Eindhoven
The Netherlands:
tel. (040) 782285.



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....

.....

.....

.....

.....

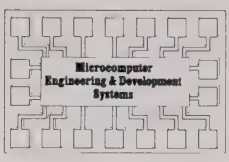
Fold here

Fold here

Seal

Please do not staple

Seal



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....

.....

.....

.....

.....

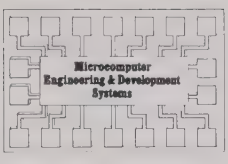
Fold here

Fold here

Seal

Please do not staple

Seal



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....
.....
.....
.....
.....

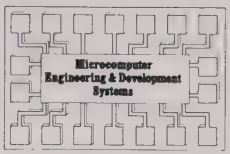
Fold here

Fold here

Seal

Please do not staple

Seal



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....

.....

.....

.....

.....

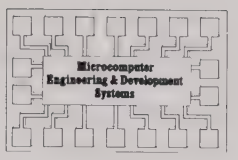
Fold here

Fold here

Seal

Please do not staple

Seal



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....

.....

.....

.....

.....

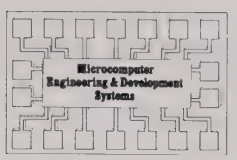
Fold here

Fold here

Seal

Please do not staple

Seal



PROBLEM REPORT/CHANGE REQUEST

Please use typewriter or write in block capitals.

PRIORITY: ☐ very urgent ☐ urgent ☐ routine

Your submissions are appreciated and contribute to further improvement of our products.
Please use one form for each problem or change request.

SUBMITTED BY: Date .....
Name .....
Company .....
Address .....
Town ..... Postcode .....
Country .....
Tel: ..... Telex .....
YOUR SUPPORT AGREEMENT NR.: .....
SYSTEM NUMBER NC ..... Serial Nr. .....

Note that we can only accept problem reports and change requests for those parts of your system which have been delivered by Philips.

CONCERNS
Operating system: ..... Release version: .....
Philips SW: ..... Release version: .....
Hardware: ..... Manual .....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ATTACHED: ☐ Diskette ☐ Listing ☐ Others:

PHILIPS USE ONLY Date Date
PR nr.: NSO: SC:

After completion please mail to your national Philips Sales and Service organisation.
4822 871 90444

PHILIPS

PLACE
STAMP
HERE

PHILIPS' REPLY CARD

Address to your national Philips Sales and Service organisation. See address list in this manual.

.....

.....

.....

.....

.....

Fold here

Fold here

Seal

Please do not staple

Seal

PHILIPS Sales and Service World-wide

Australia:
Philips Scientific & Industrial
Equipment Div. Centrecourt. 25–27 Paul
Street, North Ryde. Sydney N.S.W. 2113;
tel. 88 88 222

Austria:
Oesterreichische Philips Industrie
GmbH, Abteilung Industrie
Elektronik, Parttartgasse 32, A–1230, Wien;
tel. (0222) 841611

Belgium:
N.V. Philips & MBLE Associated S.A.
Scientific & Industrial Equipment Division
80 Rue des Deux Gares, Bruxelles;
tel. 2–5230000

Canada:
Philips Electronics Ltd.
601 Milner Avenue
Scarborough, Ontario M1B1Mb;
tel. (416) 292–5161.

Denmark:
Philips Elektronik Systemer A/S
Afd. Industri & Forskning
Strandlodsvej 4DK–2300–København S;
tel. (1) 572222; telex 31245.

Finland:
OY Philips AB, Kaivokatu 8
P.O. Box 255, 00101 Helsinki 10;
tel. 01–7271.

France:
S.A. Philips Industrielle et Commerciale
Science et Industry
105 Rue de Paris, 93002 Bobigny;
tel. 1–8301111

Germany (F.R.G.):
Philips GmbH
Unternehmensbereich Elektronik fuer
Wissenschaft und Industrie
3500, 35 Kassel; tel. 561–5010.

Great Britain:
Pye Unicam Ltd.
York Street
Cambridge CB1 2 PX;
tel. (0223) 358866.

Hong Kong:
Philips Hong Kong Ltd.
P.O.B. 2108, 27–29th floor Hopewell Centre
17 Kennedy Road, Hong Kong;
tel. 5–283298.

India:
Peico Electronics & Electricals Ltd.
PLOT 80, Bhosari Industrial Estate
Pune 411026;
tel. 82516/83547.

Ireland:
Philips Electrical (Ireland) Ltd.
Newstead, Clonskeagh
Dublin 14;
tel. 693355.

Italy:
Philips S.P.A., Sezione I&E T&M dept.
Viale Elvezia 2
20052 Monza;
tel. (039) 3635.1, telex 333343.

Netherlands:
Philips Nederland B.V.
Boschdijk 525, Gebouw VB
5600 MD, Eindhoven;
tel. (040) 782808.

New Zealand:
Philips Electrical Ind. Ltd.
Scientific & Industrial Division
68–86 Jervois Quay, P.O.Box 2097
Wellington C1; tel. 735735.

Norway:
Norsk Aktieselskap Philips
Dept. Industry and Telecommunication
Sandstuveien 70, P.O.Box 1, Mangelrud
Oslo 6 tel: 2–680200

PHILIPS Sales and Service World-wide

Portugal:
Philips Portuguesa S.A.R.L.
Av. Eng. Duarte Pacheco. Appartado 1331
Lisbon–1:
tel. (019) 683121.

Singapore:
Philips Singapore Private Ltd.
Professional Division, Lorong 1, Toa Payoh
P.O.Box 340, Toa Payoh Central Post Office
Singapore–1231:
tel. 2538811.

Spain:
Philips Iberica S.A.E.
Dpto Apparatos de Medida
Martinez Villergas 2
Madrid:
tel. 404–2200.

Sweden:
Svenska A.B. Philips
Div. Industrielektronik Avd. Matinstrument
11564 Stockholm:
tel. (08) 7821000.

Switzerland:
Philips AG, I&E Equipment
Allmendstrasse 140, Postfach 670
CH–8027 Zurich:
tel. (01) 488211.

Taiwan:
Philips Taiwan Ltd.
6th Floor, 150 Tun Hua N. Road
P.O.Box 22978, Taipei:
tel. (02) 712–0500.

U.S.A.:
Philips Test and Measurement Department Inc.
85, McKee Drive, Mahwah,
New Jersey 07430:
tel. 201–529–3800.

For countries not listed above:
Philips Export BV.
I&E Division Building HBS
P.O.Box 218,
5600 MD Eindhoven
The Netherlands:
tel. (040) 782285.

4822 871 90367